Étude et réalisation d'un émulateur pour framework d'applications mobiles hybrides

Jean-François Greffier

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL DE RENNES

MEMOIRE

présenté en vue d'obtenir

LE DIPLOME D'INGENIEUR CNAM

SPECIALITE : Informatique

OPTION : Architecture et Ingénierie des Systèmes et des Logiciels

par

GREFFIER Jean-François

Étude et réalisation d'un émulateur pour framework d'applications mobiles hybrides

Soutenu le 06/01/2017

JURY

PRESIDENT :	M. Yann POLLET	*CNAM*
MEMBRES :	M. Erwann GOUESBET	*Amdocs*
	M. Christophe MICHEL	*Amdocs*

Remerciements

Tout d’abord je remercie mon employeur Streamezzo Amdocs pour me donner la possibilité d’évoluer dans un environnement à la fois international et enrichissant techniquement.

Je tiens à remercier également mon tuteur en entreprise Christophe MICHEL, souvent mon premier lecteur, pour ses conseils avisés. Je remercie chaleureusement mon manager Erwann GOUESBET pour son soutien dans ma démarche au CNAM. Mes pensées vont aussi vers l’ensemble de mes collègues travaillant ou ayant travaillé sur UXFME.

Merci tout particulièrement à mon épouse GONG Ying pour son support sans failles pendant toutes ces années passées au CNAM et pour sa compréhension pendant la rédaction de ce mémoire.

Je suis reconnaissant envers mon tuteur CNAM Florian Pervès pour ses conseils, son soutien et sa disponibilité pendant la réalisation de ce mémoire.
Je remercie également l’ensemble du personnel du CNAM Bretagne.

Glossaire

Agilité : capacité d'une organisation à fournir tôt et fréquemment des services impactant ses utilisateurs, tout en s'adaptant à temps aux changements dans son environnement.

API : Application Programming Interface, interface de programmation applicative. Ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.

Application mobile hybride : application mobile écrite en HTML5 et rendue par une WebView au coeur d'un conteneur natif.

Backlog : dépôt de collecte et de partage des choses à faire, ou stories, qui apportent de la valeur.

CES : Customer Experience Solution, la suite Amdocs de gestion opérationnelle, réseaux et de client.

Développeur : membre de l'équipe Scrum.

DOM : le Document Object Model est une interface qui permet d'accéder et de mettre à jour le contenu, la structure et le style d'un document HTML ou XML.

Équipe pluridisciplinaire : équipe qui possède, collectivement, toutes les compétences requises pour développer le produit.

FOSS : Free and Open-Source Software, logiciel libre et open-source.

HTTP : Hyper Text Transport Protocol, protocole client-serveur pour le web.

IDE : Integrated Development Environment, environnement de développement intégré comprenant tous les outils nécessaires au développement dans un langage cible. Cela peut comprendre un éditeur de texte, un compilateur, un émulateur...

JSON : JavaScript Object Notation, format d'échange de données populaire basé sur la syntaxe du JavaScript.

Mêlée quotidienne : moment de partage quotidien entre les membres de l'équipe permettant d'organiser la journée et de vérifier l'alignement sur l'objectif du sprint.

Partie prenante : toute personne intéressée par les résultats de l'équipe.

Product Owner : personne de l'équipe ayant délégation des parties prenantes pour maximiser la valeur, est imputable du résultat auprès d'elles.

REST : REpresentational State Transfer, style d'architecture utilisé dans les services web et se reposant sur le protocole HTTP pour définir une série d'opérations possibles sur des ressources.

Scrum Master : personne au service de l'équipe, pour l'aider à réaliser les travaux demandés par le Product Owner, en appliquant Scrum dans le contexte de l'organisation.

SOAP : Simple Object Access Protocol, protocole d'appels de procédures basé sur XML.

Spike : story qui est une étude permettant d'obtenir une réponse à un problème technique.

Sprint : bloc de temps fixé aboutissant à créer un incrément du produit potentiellement livrable.

Story : élément du backlog, apportant de la valeur à une partie prenante ou à l'équipe.

SVG : Scalable Vector Graphics, format d'images vectorielles basé sur XML.

UXF : User Experience Framework, solution basée sur des frameworks HTML/JavaScript modernes permet la création rapide d'applications professionnelles s'interfaçant facilement avec les solutions back-end d'Amdocs.

UXFME : User Experience Framework Mobile Enablement, solution Amdocs pour développer, packager et maintenir des applications mobiles hybrides.

WebView : composant logiciel similaire au navigateur web, pouvant être utilisé dans une application.

XML : Extensible Markup Language, langage de balisage générique pouvant être utilisé comme format d'échange de données.

Table des matières

1 Introduction

Les appareils mobiles tels que smartphones et tablettes sont dorénavant omniprésents. En France, une étude Médiamétrie de 2015 estime que le trafic web se fait majoritairement depuis un appareil mobile[1]. Parallèlement les applications mobiles se sont complexifiées et sont devenues de plus en plus complexes à produire. Le développement d'applications mobiles hybrides, mêlant technologies web et natives, est devenu une tendance forte de l'industrie. Par ailleurs les outils d'aide au développement d'applications sont des atouts sérieux pour les entreprises qui ne peuvent plus négliger les plateformes mobiles, l'approche hybride leur permettant notamment de cibler plusieurs de ces plateformes sans requérir à un développement spécifique pour chacune, induisant un surcoût en temps, ressources et budget.

Ce mémoire décrit les résultats de l'étude et de la réalisation d'un émulateur pour le framework UXFME, solution développée par Streamezzo Amdocs dédiée au développement d'applications mobiles hybrides.

L'entreprise Streamezzo Amdocs, son historique et ses activités sont liés à l'industrie des télécommunications en général, et au mobile en particulier. C'est pourquoi après sa présentation un rappel sur le marché actuel des applications mobiles sera fait, ainsi qu'une présentation des différents types d'applications. Sera ensuite détaillée la suite logicielle UXFME, solution dédiée au développement d'applications mobiles hybrides. L'équipe UXFME, ses méthodes et son organisation seront introduits.

Le besoin initial et la motivation d'un tel émulateur seront ensuite décrits. L'étape d'étude comprendra un tour d'horizon des solutions existantes, leurs avantages, leurs inconvénients et leur pertinence dans le cadre du projet. Cette partie consacrée à l'étude expliquera également la conception du logiciel et les choix architecturaux qui ont été réalisés.

Dans une troisième partie sera présenté comment la solution découlant de la phase d'étude est alors réalisée. Les choix techniques en fonction de l'étude, de la faisabilité

[1] Les chiffres clés de la mesure web analytics de Médiamétrie, communiqué de presse, Sept. 2015

technique et de l'environnement de l’entreprise seront détaillés, ainsi que les points durs de la mise en oeuvre de l’émulation en elle-même, de même que la réalisation de l’interface utilisateur correspondante.

Enfin, un bilan du projet sera effectué : les difficultés rencontrées, et les enseignements qui peuvent en être tirés. Le cycle de vie de l’application, sa maintenance et son évolution seront présentés avant de proposer une conclusion et des perspectives pour le futur de cet émulateur et plus généralement des outils d’aide au développement d’applications mobiles hybrides.

2 Contexte

Cette première partie détaille le contexte du projet d'émulateur. Tout d'abord l'entreprise Streamezzo, son historique depuis ses débuts jusqu'au rachat par Amdocs, et ses activités au sein de ce groupe international. Une présentation des principes régissant les applications mobiles hybrides permettra d'éclairer le lecteur sur les enjeux du développement d'applications mobiles et l'environnement technique du projet. Plus spécifiquement sera détaillée la suite logicielle UXFME, solution de Streamezzo Amdocs dédiée au développement d'applications mobiles hybrides. Enfin, l'organisation et la méthodologie de l'équipe développant UXFME seront présentées, car elles ont un impact direct sur la façon de mener l'étude et la réalisation d'un projet.

2.1 Streamezzo Amdocs

2.1.1 Streamezzo

Streamezzo est une entreprise française résultant d'un essaimage de France Télécom en 2004. Elle se spécialise alors dans la création d'application riches pour téléphones portables et de solutions pour développer, déboguer et déployer des services multimédias. Sa technologie "Rich Media" permet notamment la création d'applications de télévision mobile, de widgets ou de portails multimédia. Ses clients sont alors les opérateurs mobiles, les constructeurs de téléphones ou les fournisseurs de contenus et de services. La société est déjà majoritairement tournée vers l'international avec des clients en Allemagne, Australie, Chine, Israël, Russie…
En 2010, Streamezzo fait l'objet d'une acquisition de la part de la société Amdocs.

2.1.2 Amdocs

Amdocs est une entreprise américaine fondée en 1982 en Israël. Amdocs développe et fournit des solutions de gestion opérationnelle, réseaux et de gestion client pour les grands opérateurs mobiles et de télécommunication. L'entreprise cotée au NASDAQ (DOX) compte plus de 24 000 employés dans le monde et a dégagé un chiffre d'affaire de 3.6 milliards de dollars en 2015[2].

Sa principale offre est Customer Experience Solution (CES) un large portfolio d'applications à destination des opérateurs mobiles, câble ou fixe. Cela inclut par exemple des systèmes de soutien fonctionnel aux entreprises (Business Support System) permettant la gestion de client, de commande, de facturation, de tarification. On peut noter aussi des solutions du domaine opérationnel (Operational Support System), mais également d'optimisation réseau, de paiement mobile, de facturation etc...

Figure 1 : Représentation géographique de clients Amdocs

Amdocs sert plus de 250 clients dans 90 pays. C'est directement ou indirectement plus de deux milliards d'utilisateurs finaux qui utilisent une solution Amdocs de façon quotidienne.

[2] Rapport annuel 2015 http://www.amdocs.com/Documents/AnnualReport2015.pdf

2.1.3 Acquisition

A partir des années 2000, Amdocs fait régulièrement l'acquisition d'entreprises pour assurer sa croissance.

Tableau 1 : Acquisitions d'entreprises par Amdocs 2000-2015[3]

Année	Acquisition
2015	Comverse Inc.
2013	Actix ; Celcite
2011	Bridgewater Systems
2010	MX Telecom ; **Streamezzo**
2009	jNetX
2008	Jacobs Rimell Ltd. ; Changingworlds
2006	Qpass ; Cramer Systems
2005	DST Innovis
2003	XACCT Technologies
2000	Solect Technology Group

Streamezzo est ainsi devenue l'entité spécialisée dans le développement mobile et l'outillage pour la création d'applications mobiles d'Amdocs et de ses clients. Elle est impliquée dans le développement d'applications mobiles de relation client, de tableau de bord, de self-service. Streamezzo est également au cœur de la solution de domotique Amdocs Connected Home. L'entité participe au développement de Matrix, un protocole de communication open-source dédié à la messagerie instantanée mais aussi à l'Internet des objets.

Enfin, Streamezzo assure la conception et le développement d'une suite logicielle dédiée au développement et au déploiement d'applications mobiles hybrides, UXFME, et fournit du support aux équipes réalisant des applications mobiles hybrides s'appuyant sur cette solution.

[3] https://www.crunchbase.com/organization/amdocs/acquisitions

2.2 L’industrie des applications mobiles

Les smartphones et tablettes ont un impact majeur sur l’industrie logicielle dans son ensemble, que ce soit dans les usages des utilisateurs ou pour ses acteurs, maintenant majoritairement tournés vers le mobile. Les applications mobiles, maintenant principalement distribuées par l’intermédiaire d’un magasin d’applications (app store), sont donc extrêmement importantes. C’est particulièrement le cas pour les clients d’Amdocs tels que les câblo-opérateurs, les opérateurs mobiles ou les fournisseurs Internet qui se doivent d’offrir à leurs utilisateurs des solutions logicielles mobiles. Par exemple, une forte tendance est au selfcare qui permet aux clients de gérer eux-mêmes leur compte mobile ou Internet (forfait, consommation, options…) ou bien de régler leurs problèmes d’ordre technique. Amdocs doit donc être en mesure de développer des applications mobiles répondant aux attentes de ses clients et des utilisateurs finaux.

2.2.1 Parts de marché

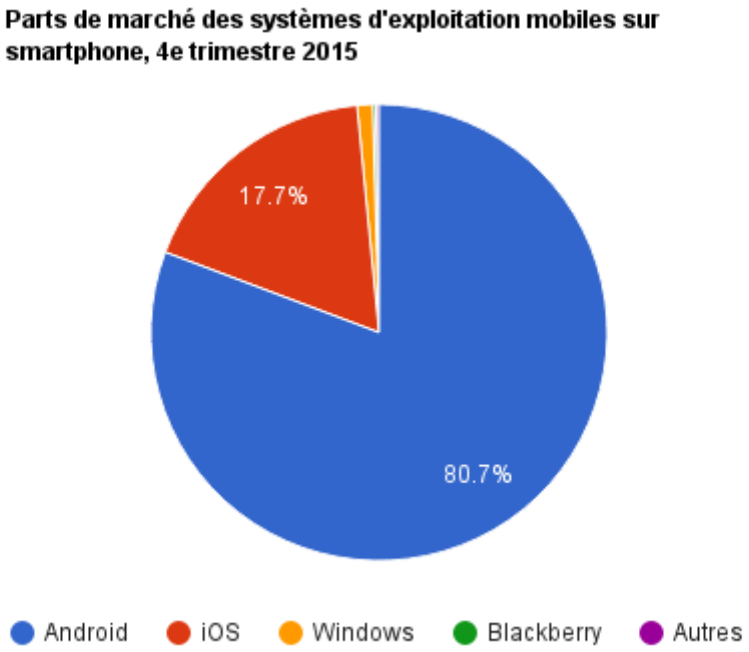


Tableau 2 : Ventes mondiales de smartphones par système d'exploitation au 4e trimestre 2015, Gartner[4]

Système d'exploitation	Milliers d'unités 4e trimestre 2015	Part de marché 4e trimestre 2015
Android	325394.4	80.7%
iOS	71525.9	17.7%
Windows	4395	1.1%
Blackberry	906.9	0.2%
Others	887.3	0.2%
Total	403109.4	

Le système d'exploitation Android, le plus populaire, compte en 2015 quelque 1.4 milliards d'utilisateurs actifs[5]. Le système d'exploitation iOS, utilisés sur les appareils mobiles Apple (iPhone et iPad), vient en deuxième position et reste néanmoins une plateforme très importante pour les développeurs d'application. En effet, et comme exposé dans la section suivante, les utilisateurs d'iOS dépensent en moyenne plus que les autres.

Cette domination au niveau mondial des deux premiers acteurs cache une disparité selon les régions géographiques. Par exemple les systèmes d'exploitation Microsoft Windows (Windows Phone ou Windows 10 Mobile) sont très populaires en Amérique du Sud.

C'est donc pour cela qu'une application mobile se doit d'être multi-plateforme, en visant à minima Android et iOS.

[4] http://www.gartner.com/newsroom/id/3215217

[5] http://www.androidcentral.com/google-says-there-are-now-14-billion-active-android-devices-worldwide

2.2.2 Les magasins d'applications

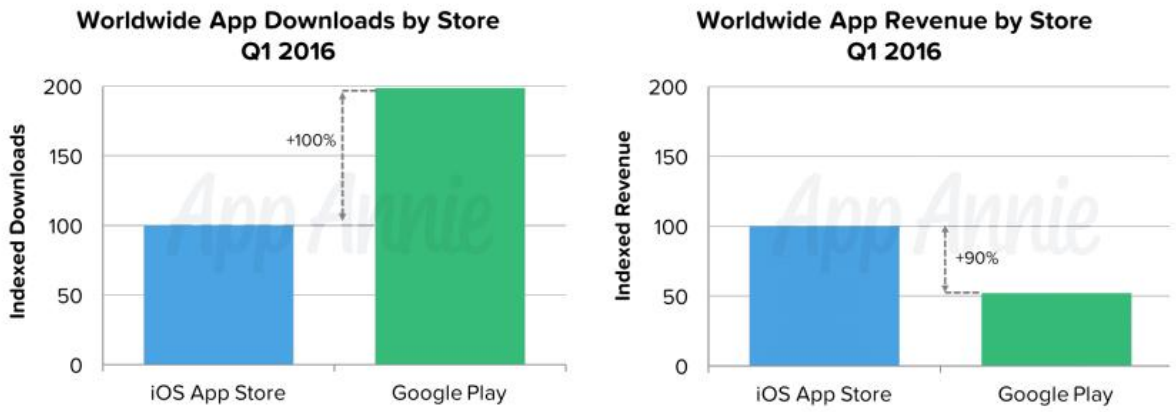


Figure 2 : Téléchargements d'applications en millions, et revenus en millions de dollars sur les magasins Apple et Google, rapport App Annie[6]

Depuis l'émergence de l'iPhone avec son App Store et d'Android avec le Google Play Store la même année[7], le magasin d'application est devenu le standard *de facto* pour la distribution d'application mobiles. Il s'agit le plus souvent d'une boutique en ligne accessible depuis un site web et/ou d'une application dédiée sur l'appareil mobile qui permet de télécharger et d'installer des logiciels directement sur son smartphone ou sa tablette.
Les millions d'applications distribuées par ce biais engendrent des revenus colossaux, partagés entre le distributeur et les éditeurs de logiciels. On notera que même si la base d'utilisateurs et le nombre d'application est plus élevé sur la plateforme Google Play, c'est l'App Store d'iOS qui génère le plus de revenus.

Pourtant, la distribution par ce biais est contraignante pour un éditeur logiciel. En effet, le développeur doit accepter les conditions de distribution du magasin d'application et soumettre l'application à plusieurs vérifications. Cela va de la mise en conformité avec les réglementations locales à des considérations sur l'expérience utilisateur ou les performances de l'application. Ce processus largement laissé à la discrétion du fournisseur du magasin d'applications, notamment Apple, implique notamment une expérimentation préalable de l'application par un acteur humain, qui jugera de la conformité de l'application mobile soumise avec les règles de publication en vigueur.

[6] http://go.appannie.com/report-app-annie-index-market-q1-2016
[7] 10 juillet et 22 octobre 2008

Cette procédure est complexe techniquement et longue, le distributeur mettant généralement plusieurs jours à donner un retour, ceci s'expliquant notamment par l'implication d'un acteur humain, le testeur. Par ailleurs, certaines règles peuvent être implicites ou mal documentées et ainsi engendrer de nombreux échanges avant la publication finale sur le magasin d'applications.

2.2.3 Les applications mobiles hybrides

Il existe plusieurs approches pour créer une application mobile, avec leurs propres avantages et inconvénients. Les applications mobiles peuvent être de trois types : native, web ou hybride. Ces trois types sont présentés ci-après.

Une application dite native est une application qui a été développée dans le langage dédié à la plateforme visée. Par exemple pour les principales plateformes mobiles : Java pour Android, Objective C ou Swift pour iOS, C# pour Windows Phone. En plus d'avoir à appréhender des langages et des environnements de développement différents, le développeur doit maîtriser les APIs spécifiques de chaque système d'exploitation.
L'application peut alors être distribuée par un magasin d'applications, qui est devenue la méthode standard *de facto*. L'apparence et l'ergonomie est celle de l'appareil mobile, et l'ensemble des fonctionnalités des APIs est accessible aux développeurs. Enfin, les performances sont optimales.
La contrepartie est que le développement d'applications natives est complexe et coûteux. Dans le cas d'application mobiles multi-plateforme, il faut multiplier les équipes de développement et les coordonner pour pouvoir offrir la même expérience pour tous les utilisateurs. Enfin, la publication dans un magasin d'application est contraignante de par les délais que cela comprend. C'est particulièrement problématique en cas de mise à jour, puisqu'il faudra alors soumettre l'application à nouveau.

Une application web est une application riche accessible depuis un navigateur Internet. Elle est développée en s'appuyant sur les technologies web HTML5, JavaScript et CSS.

L’application est directement accessible depuis un navigateur web, ce qui n’est pas la pratique standard pour une application dans le monde mobile, le risque étant que l’application perde de sa visibilité pour les utilisateurs finaux qui sont habitués à découvrir, acheter, et installer des applications depuis un store. L’apparence de l’application n’est pas par défaut celle inhérente à l’appareil. Cependant, il existe de nombreux frameworks qui offrent des outils pour se rapprocher du style natif en termes d’expérience utilisateur. Même si les fonctionnalités sont plus riches grâce à HTML5 (vidéo, ...) certaines sont toujours absentes. Les performances s’améliorent mais restent toujours en deçà du natif, par exemple pour les applications les plus exigeantes comme les jeux vidéo.
Le problème de la fragmentation des plateformes mobiles se limite dans ce cas à celui des navigateurs, problème bien connu des développeurs web. La complexité et le coût du multi-plateforme est donc réduit. De plus, une application web est toujours à jour si le contenu est distant (pas de problématique de publication dans un store dans ce cas). Il est également possible de mettre en cache du contenu et donc de faire des applications hors-ligne qui se mettent à jour totalement ou partiellement.

Une application mobile hybride est composée d’un compromis entre les deux approches précédentes : c’est une application écrite en HTML5 rendue par une WebView, un composant proche du navigateur web, au coeur d’un conteneur natif.
Comme une application native, elle est distribuée via un magasin d’applications. L’apparence est celle définie dans la partie web, qui peut donc se rapprocher des standards de la plateforme. Les fonctionnalités des APIs sont accessibles aux développeurs web si la partie native les expose, par exemple dans le cadre d’un framework d’applications mobiles hybrides. Les performances sont celles de la WebView, donc proches ou identiques à celles du navigateur web.
La complexité pour le développeur web est abstraite par le framework d’applications mobiles hybrides. En effet, le logiciel est développé comme une application web, avec des APIs spécifiques pour accéder aux fonctionnalités natives. La complexité et le coût du multi-plateforme est donc réduit. Enfin, il est possible de distribuer une application mobile hybride via un magasin d’applications, tout en gardant la possibilité de mettre à jour son contenu web de façon indépendante.

Tableau 3 : Différences entre natif, web et hybride. Point de vue de l'utilisateur final

	Native	Web	Hybride
Distribution	✓	✗	✓
Apparence	✓✓ excellent	✓	✓
Fonctionnalités	✓	✗	✓
Performances	✓✓ excellent	✓	✓

Tableau 4 : Différences entre natif, web et hybride. Perspective du développeur ou éditeur de logiciel

	Native	Web	Hybride
Complexité	✗	✓	✓
Coût	✗	✓	✓
Multi-plateforme	✗	✓	✓
Mise à jour	✗	✓	✓✓ excellent

Avec l'approche hybride, une application mobile peut offrir une expérience utilisateur satisfaisante sur plusieurs plateformes à un coût plus faible. Une telle application mobile hybride peut être créée plus rapidement, plus facilement, et être plus facilement maintenue.

Il faut pour cela une solution complète qui permet le développement d'applications mobiles hybrides mais aussi leur packaging, et la gestion de leur cycle de vie sur l'ensemble des plateformes mobiles significatives : Android, iOS, Windows Phone et Windows 10. C'est ce que propose le framework UXFME.

2.3 UXFME

2.3.1 Présentation

Avec l'apparition du HTML5 comme solution de développement d'applications, Amdocs a créé un socle logiciel nommé User eXperience Framework (UXF). Cette solution basée sur des frameworks HTML/JavaScript modernes permet la création rapide d'applications professionnelles s'interfaçant facilement avec les solutions back-end d'Amdocs.
La solution UXF se compose ainsi à la fois d'une partie cliente (widgets graphiques configurables) et serveur (connecteurs sous forme d'API REST).

User eXperience Framework Mobile Enablement (UXFME) est une solution pour créer des applications mobiles hybrides. UXFME permet de créer des applications mobiles développées en HTML5 pour les principales plateformes : Android, iOS, Windows Phone 8, Windows Phone 8.1, Windows 10 (pour tablettes) et Windows 10 Mobile. UXFME était initialement prévu pour être un complément à UXF afin de couvrir les plateformes mobiles, mais peut être utilisé avec n'importe quel contenu HTML/JavaScript/CSS (y compris autre que UXF).

UXFME est un outil **fondation** du portfolio Customer Experience Solution d'Amdocs. C'est-à-dire que cette solution interne est privilégiée par rapport à d'autres quand une nouvelle application mobile est créée, permettant ainsi une harmonisation des briques logicielles utilisées dans l'architecture des produits fournis par Amdocs.

Figure 3 : Développement et publication d'une application avec UXFME

La solution UXFME permet de développer, packager et maintenir des applications mobiles hybrides professionnelles pour toutes les plateformes mobiles, s'appuyant sur des standards web et bénéficiant d'une intégration fine avec les fonctions des appareils mobiles.

Il en résulte des délais de commercialisation plus court pour ces applications mobiles hybrides, adressant un marché de masse, et permettant d'apporter des mises à jour et évolutions de façon intelligente aux applications une fois distribuées aux utilisateurs finaux.

2.3.2 Architecture

UXFME est un framework se présentant sous la forme de de nombreux composants, présentés ci-après.

UXFME Build Service

Le "Build Service" est un logiciel permettant la configuration et le packaging d'application mobile hybride. Il offre une API Node.js facilitant l'intégration aux outils usuels de compilation d'application web, comme par exemple Grunt ou Gulp. L'outil est disponible à la fois sous forme d'un formulaire web et d'un module utilisable par API et ligne de commande (deux modules Node.js distincts).

UXFME Build Platform

La création d'application est ensuite déléguée à une plateforme distante : UXFME Build Platform. Cette plateforme est exposée sous la forme d'un service web REST permettant de créer des applications mobiles hybrides. Les applications sont construites pour chaque plateforme demandée : iOS, Android, Windows Phone 8, Windows Phone 8.1, Windows 8.1, Windows 10. Ainsi le développeur souhaitant créer une application hybride mobile n'a pas à se préoccuper de l'installation, de la configuration et de la maintenance d'une telle plateforme de construction d'application, à la fois complexe, hétérogène et sujette à de fréquentes mises à jour, notamment pour suivre l'évolution des versions des systèmes d'exploitation mobiles mis sur le marché. UXFME Build Platform permet par exemple de s'abstraire du problème du système d'exploitation nécessaire pour construire une application mobile. En effet, il faut impérativement un Apple Mac pour créer une application pour iPhone ou iPad, une machine sous Windows 10 pour Windows 10 Mobile, etc...

UXFME Client Runtime

Le client UXFME offre des Device APIs pour interagir avec les capacités de l'appareil mobile depuis l'application mobile hybride. Il fait partie intégrante de l'application hybride mobile packagée grâce aux composants précédemment présentés.

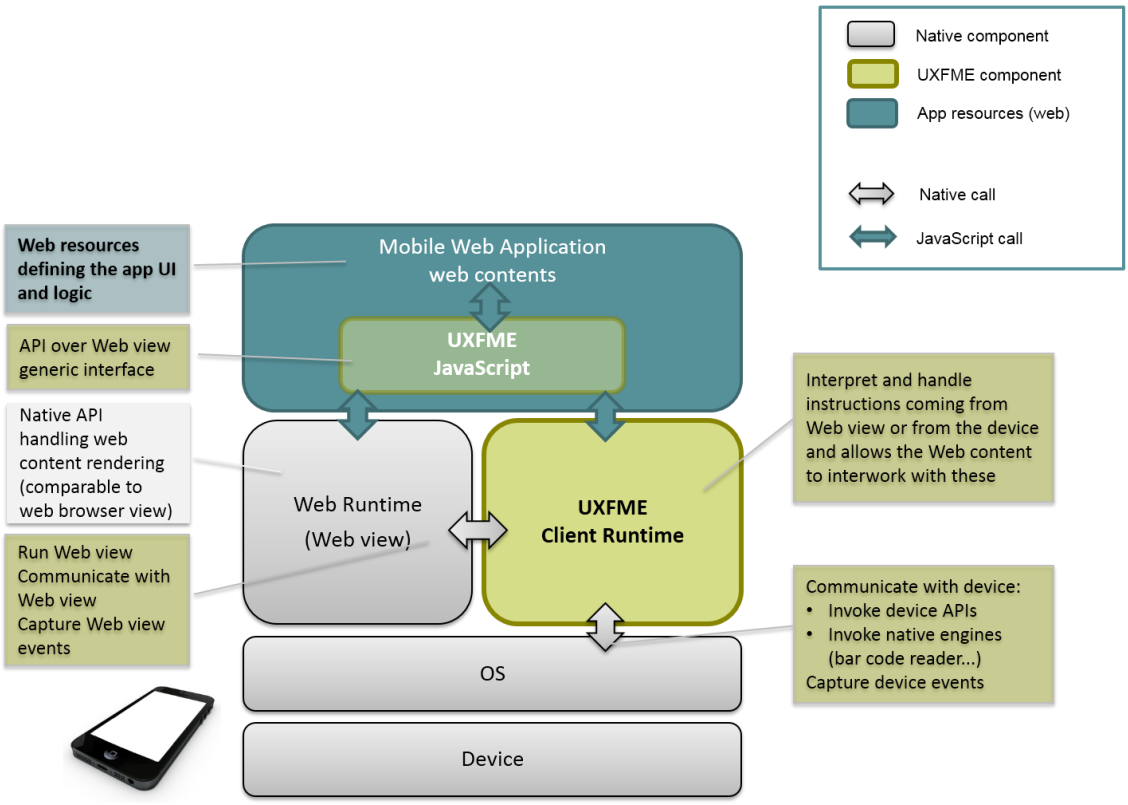


Figure 4 : Architecture haut niveau du client UXFME

En complément, les applications web embarquent et utilisent la bibliothèque JavaScript UXFME, exposant ces APIs sous la forme d'une signature JavaScript unifiée pour toutes les plateformes mobiles supportées.
L'application alors obtenue repose sur le composant WebView de l'appareil mobile et sur le client UXFME. C'est ce dernier qui assure l'abstraction des différents systèmes d'exploitation et matériels des appareils mobiles.

De plus des fonctions de sécurisation de l'application sont aussi rendues possibles par le client UXFME. Par exemple la signature ou le chiffrement des ressources web de l'application, la protection des données sensibles de l'application, l'identification des serveurs de confiance accédés par l'application.

Enfin, le client UXFME offre plusieurs outils pour le développement et le débogage tels que des logs et un mode debug activable. L'émulateur UXFME a été développé pour enrichir cette offre d'outils destinés aux développeurs d'applications mobiles hybrides.

UXFME Applications Manager

Ce composant prend en charge la gestion du cycle de vie des applications avec la mise à disposition d'applications (téléchargement sur les appareils mobiles) et les mises à jour.

UXFME Push Notification Service

Le service de Notification Push d'UXFME permet l'envoi de notifications aux applications clientes préalablement enregistrées sur ce service, et couvre l'ensemble des plateformes mobiles supportées par UXFME.
Une notification push est un message court envoyé à un appareil mobile, et affiché sur l'interface native de l'appareil. Il peut aussi transmettre des données à l'application afin qu'elle puisse alors réagir en conséquence.

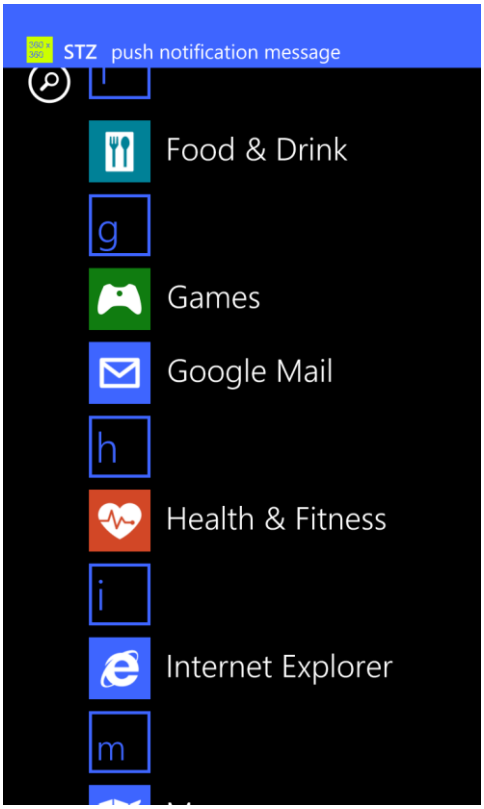


Figure 5 : Notification Push sur Windows Phone 8.1

Le Push Notification Service assure l'abstraction des systèmes de notifications d'Apple, Google et Microsoft, permettant ainsi aux développeurs d'applications d'envoyer facilement une notification à n'importe quel appareil mobile, au travers d'une unique API REST exposée.

UXFME Device Certification

Le processus de “Device Certification” consiste à tester, adapter puis certifier des appareils mobiles choisis selon la part de marché qu’ils représentent et les demandes clients. La certification se déroule en trois étapes :

1. On procède à une qualification qui consiste à tester le client UXFME, les Devices APIs ainsi que le niveau de support des fonctionnalités HTML5 d’un appareil mobile.
2. Les éventuels problèmes rencontrés durant les tests sont alors corrigés quand c’est possible. Un rapport détaillant les résultats est ensuite fourni.
3. Enfin les appareils mobiles sont certifiés et donc officiellement supportés. UXFME garantit alors le support et la maintenance du UXFME Client Runtime avec un contrat de niveau de service défini (SLA, service-level agreement).

Le rapport produit à l’issue d’une certification UXFME permet à la fois au développeur d’anticiper des problèmes lors du développement de l’application et d’aider à la résolution d’anomalies. Enfin, la certification fournit de précieuses informations sur les caractéristiques de l’appareil mobile testé.

Vue d’ensemble

La figure suivante illustre l’ensemble des composants UXFME mis en jeu à la fois pour le packaging d’une mobile mobile hybride, et pour la gestion de son cycle de vie (distribution, mise à jour, notifications push).

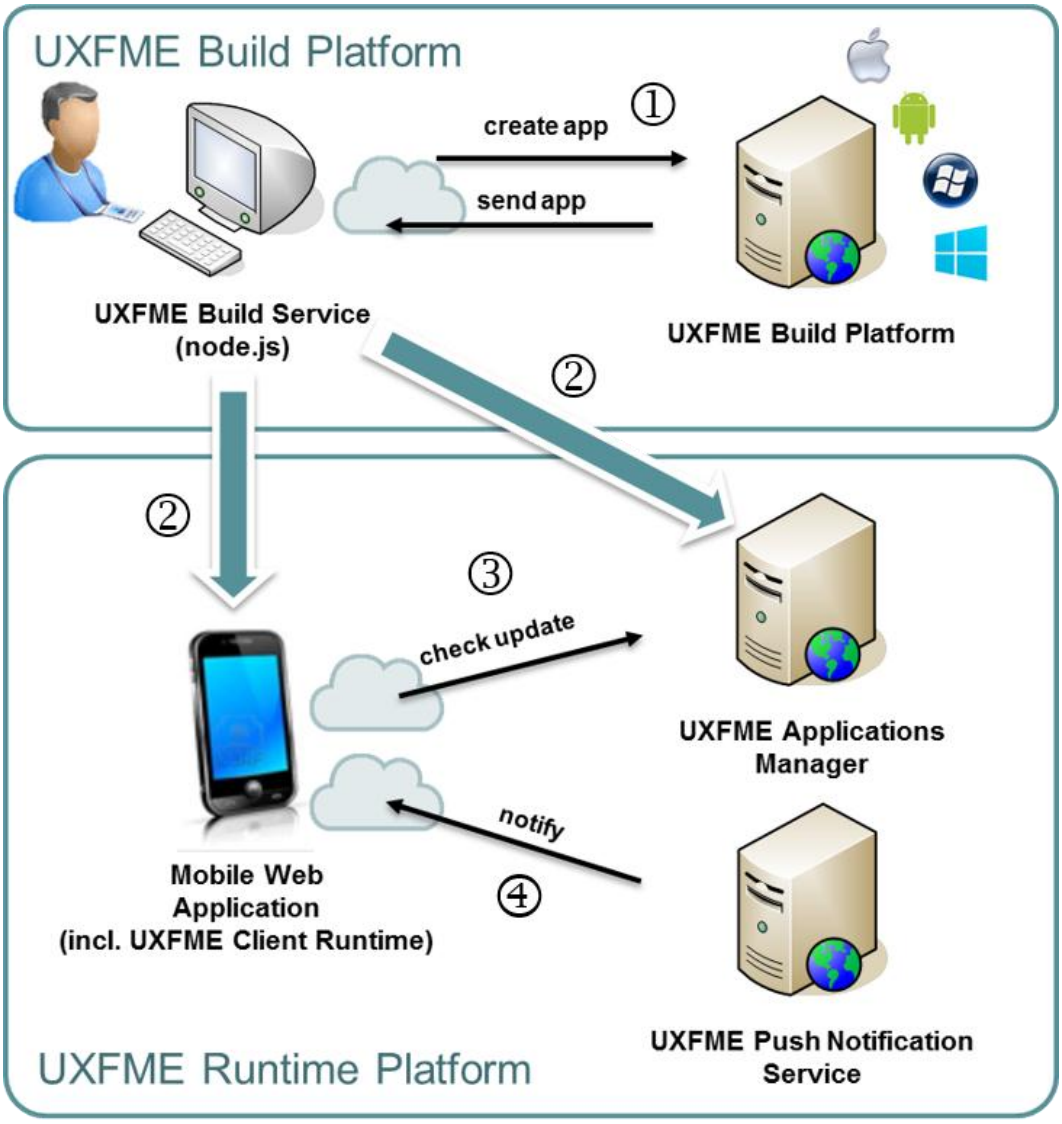


Figure 6 : Vue d'ensemble d'UXFME

Les différentes étapes mise en jeu sont les suivantes :

1. Le développeur créé une application et demande sa création via le UXFME Build Service à la Build Platform.
2. Cette application peut ensuite être déployée sur l'UXFME Applications Manager et/ou directement installée sur un appareil mobile.
3. L'application mobile pourra alors demander et éventuellement recevoir des mises à jour via le gestionnaire d'applications.
4. Le gestionnaire de notification Push pourra envoyer des notifications à l'application.

2.4 Équipe UXFME : Méthodologie et organisation

L'équipe réalisant la suite logicielle UXFME est de taille réduite, pluridisciplinaire et s'appuie sur la méthode Scrum présentée ci-après.

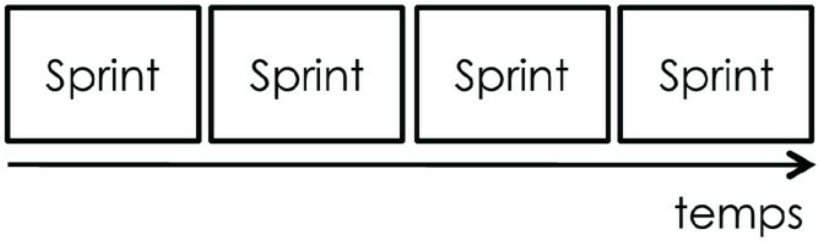
2.4.1 Scrum

Scrum est une méthode qui propose des rôles et un cadre pour la gestion de projet. Elle permet, selon ses auteurs, de répondre à des problèmes complexes dans un environnement changeant, tout en livrant des produits de qualité à un rythme régulier. Scrum s'inscrit dans le mouvement agile. Initié par le Manifeste pour le développement Agile de logiciels [1] qui définit quatre valeurs fondamentales, ce mouvement se veut pragmatique. Les méthodes agiles mettent par exemple en avant la collaboration avec les clients (plutôt que la négociation contractuelle) ou privilégie une grande réactivité aux changements demandés (plutôt que le suivi d'un plan).

Le vocabulaire Scrum est tiré du rugby, sport où l'équipe est mise en avant et où la mêlée ("scrum" en anglais) est la phase de jeu permettant de repartir sur une base solide avec un bloc constitué de joueurs côte à côte dans un même effort collectif. Plutôt que de proposer une nouvelle traduction en français des termes de Scrum, ce document suit les pratiques de l'industrie. Certains termes sont donc traduits en français mais pas d'autres.

Cycle de développement

Certains cycles de développements plus anciens comme le cycle en V décrivent des activités sans notion de temps pour chaque phase. De là il est possible de faire une analyse des besoins, des spécifications, la conception et prendre en charge le développement d'un projet pendant des mois voire des années sans qu'il n'y ait de livrable fonctionnel.



Pour éviter cela, Scrum a une approche itérative. Le sprint est la boite de temps rythmant le développement ; il est d'une durée régulière et permet de délivrer un produit ou un de ses composants régulièrement. Ainsi, les parties prenantes du projet ont une itération du produit utilisable régulièrement, à chaque fin de sprint. Néanmoins, Scrum ne définit pas les activités au sein d'un sprint pour arriver au résultat.

A chaque fin de sprint, le produit est amélioré de façon incrémentale en ajoutant des fonctionnalités, en prenant en compte le plan initial mais aussi les retours utilisateurs sur le produit existant.

Scrum impose donc un cycle de développement itératif et incrémental basé sur un cycle qui se répète successivement : le sprint.

Le backlog est une liste de choses à faire, souvent sous la forme de story.

La story, ou récit utilisateur, est un petit morceau de fonctionnalité visible d'un utilisateur [2]. Elle est décrite simplement et précisément sous la forme d'une phrase répondant à Qui ? Quoi ? (optionnellement pourquoi ?)
Par exemple :
En tant que nouvel utilisateur, je veux créer un compte qui me permettra de m'authentifier dans le système à la prochaine connexion.

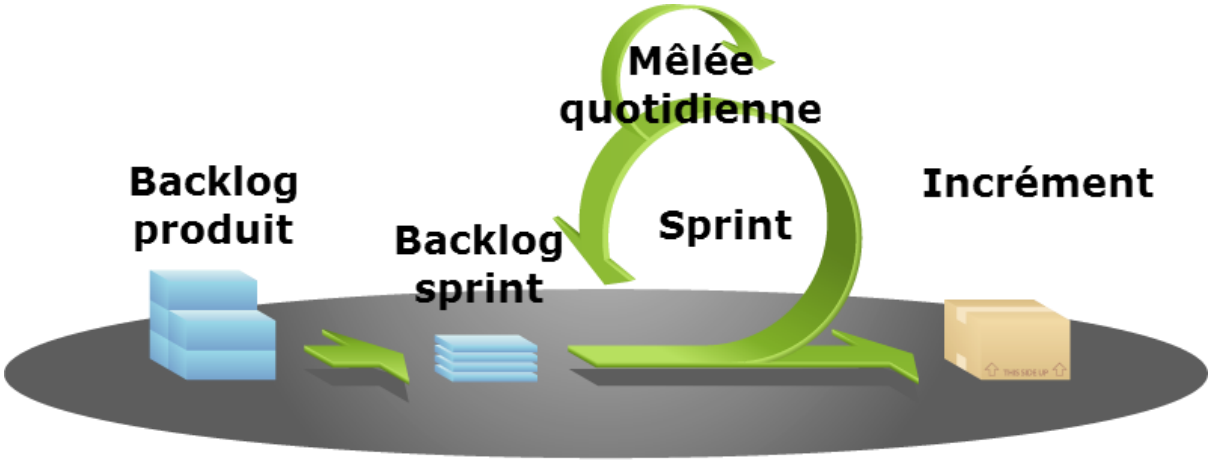


Figure 7 : Cycle de vie Scrum

Un sprint a une durée de deux à quatre semaines. En début de sprint, on fixe ses objectifs en établissant le backlog du sprint à partir d'éléments du backlog produit, en fonction des priorités et de la capacité de l'équipe. L'équipe travaille alors sur les éléments du backlog et ne livrera que les stories finies. La mêlée quotidienne est une réunion rapide où chaque membre de l'équipe évoque ses progrès, le travail à réaliser dans la journée à venir et les problèmes rencontrés. Cette réunion permet d'avoir un état des lieux précis du progrès et permet de signaler les obstacles au plus tôt. A la fin du sprint, le nouvel incrément du produit est livré et démontré aux parties prenantes. Les retours sont notés et alimenterons le backlog du produit, avant de débuter un nouveau sprint.

Rôles

Il existe trois rôles au sein d'une équipe Scrum :

- **Développeur** C'est le cœur de l'équipe. Le développeur est au sens large un membre de l'équipe qui contribue au produit. Cela peut donc être un développeur informatique mais aussi un testeur, un architecte, un data-scientist…
- **Scrum Master** Il n'est pas manager comme on pourrait le penser. C'est simplement un facilitateur. Il aide à éliminer les obstacles qui peuvent bloquer ou ralentir l'équipe, quel que soit leur nature. Le Scrum Master aide aussi à appliquer Scrum. Ce n'est pas nécessairement un rôle à temps plein, selon l'expérience de Scrum de l'équipe et son organisation. Un membre de l'équipe peut donc être à la fois développeur et Scrum Master.

- **Product Owner** C’est la personne qui est en contact avec les parties prenantes et donne une orientation au produit. Il le fait en définissant les stories du backlog et en affinant celui-ci. C’est traditionnellement un rôle à temps plein et une même personne ne peut pas assurer à la fois les rôles de Scrum Master et de Product Owner.

Viennent s’ajouter à cela des acteurs externes à l’équipe.
Les parties prenantes sont toutes les personnes intéressées par les résultats du projet [2], telles que, par exemple, le manager, le client, les autres équipes impactées ou l’utilisateur final. Si les parties prenantes sont impliquées dans le processus Scrum, elles doivent participer à la revue en fin de sprint. Elles pourront donner leurs retours et aider ainsi à améliorer la qualité du produit.
L’expert est une personne extérieure à l’équipe qui peut apporter ponctuellement son aide sur un sprint. Il peut être un expert technique ou un expert fonctionnel qui permettra de mieux comprendre l'environnement du produit et sa logique métier.

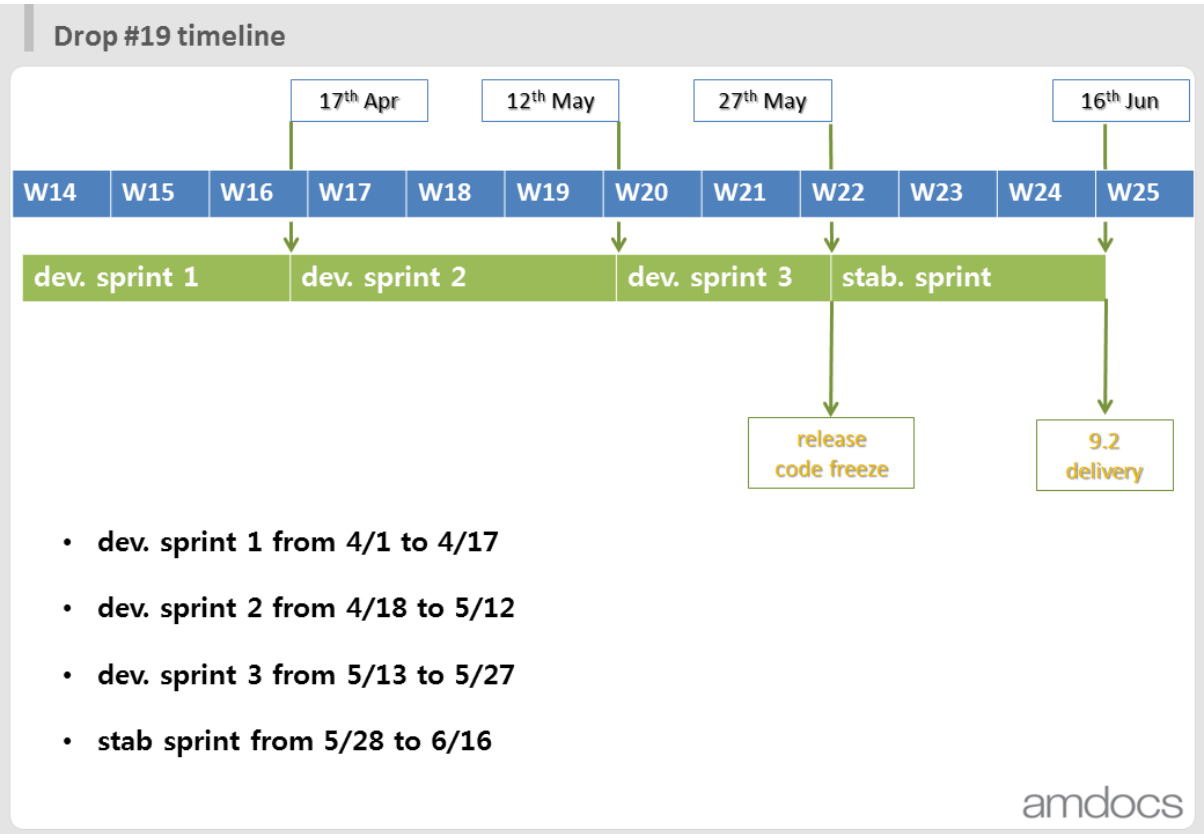
Implémentation de Scrum par l’équipe UXFME

Bien qu’adoptant Scrum pour le développement et la maintenance de l’ensemble d’UXFME, l’équipe a adapté la méthode aux besoins spécifiques du produit et de son contexte. C’est par ailleurs une démarche encouragée par la méthode.

Premièrement, les méthodes agiles privilégient des logiciels opérationnels plutôt qu’une documentation exhaustive [1]. UXFME étant une suite logicielle dédiée au développement d’applications, la documentation destinée aux utilisateurs est un livrable essentiel. Tout d’abord parce que les descriptions d’APIs destinées aux développeurs se doivent d’être exhaustives et à jour. Ensuite, la documentation détaillant l’installation et la mise à jour de la runtime platform doivent être aussi détaillée que possible. En effet, les équipes opérant sur ces plateformes peuvent être le client final lui-même ou bien l'unité dédiée à l’opérationnel au sein d’Amdocs.
Dans le cadre d’UXFME, la documentation est donc aussi importante que les fonctionnalités livrées.

Deuxièmement, Scrum préconise des livraisons rapides et régulières pour avoir rapidement des retours des parties prenantes. Le rythme de livraison pour les produits

Amdocs, et donc des fondations, est très différent. Au cycle de développement proposé par Scrum, l'équipe UXFME ajoute la notion de drop afin de livrer le produit au rythme et au niveau de qualité attendu par Amdocs. Le drop est constitué d'un ou plusieurs sprints de développement et de stabilisation. Durant le sprint de stabilisation final, la documentation utilisateur est également mise à jour et tous les livrables sont rendus publics.



L'exemple ci-dessus illustre la planification d'un drop comprenant trois sprints de développement et un sprint de stabilisation. La fin du sprint de stabilisation coïncide avec la livraison d'UXFME pour CES 9.2 (Customer Experience Solution), une version majeure de la solution d'Amdocs.

Troisièmement, il y a parfois dans la planification des stories longues qui courent sur plusieurs sprints. C'est par exemple le cas du support d'une nouvelle plateforme mobile.

Par exemple pour le support de Windows 10 :

- Ajustements dans UXFME Client Runtime / couche web, afin de prendre en charge Windows 10, exposant notamment les APIs natives
- Création du UXFME Client Runtime pour Windows 10 offrant notamment les accès aux APIs natives
- Enrichissement de la UXFME Build Platform afin d'intégrer le packaging des applications Windows 10
- Exposition des points de configuration de Windows 10 dans le UXFME Build Service
- Adaptation de la UXFME Runtime Platform, afin d'intégrer le support du provisioning d'applications Windows 10, leur mise à jour et l'envoi de Notifications Push
- Documentation utilisateur

Il n'est pas possible de livrer aux utilisateurs une partie de la chaîne de build, ou une plateforme mobile n'exposant pas l'ensemble des APIs UXFME. De fait, la finalisation et la livraison de ces stories ne peut être effectué qu'une fois l'ensemble des fonctionnalités disponibles.

Enfin, comme détaillé dans le chapitre suivant, pour développer l'ensemble des fonctionnalités, l'équipe en charge d'UXFME est pluridisciplinaire. Les membres de l'équipe ont leur domaine d'expertise respectif, ce qui permet de remplir collectivement les objectifs fixés.

2.4.2 Équipe

De la taille d'une équipe

La taille d'une équipe peut avoir une grande influence sur son efficacité. Des recherches montrent que le nombre idéal est autour de 7 [2]. C'est d'ailleurs une grande tendance dans les entreprises technologique ces dernières années.

> "Si vous ne pouvez pas nourrir une équipe avec deux pizzas, c'est que l'équipe est trop grande."[8]
> Jeff Bezos, fondateur et PDG d'Amazon

On peut citer les avantages suivants à une équipe de taille réduite :

- meilleure communication entre tous leurs membres
- tous les membres sont utiles et impliqués
- motivation accrue pour les membres

L'inconvénient principal étant que l'absence d'un membre a beaucoup plus d'impact.

Cela peut être compensé par la prise en compte des absences prévues (congés, jours fériés, formations...) dans le planning du drop. Il faut également veiller à ce que les connaissances soient diffusées à tous les membres de l'équipe.

Durant la période décrite dans ce mémoire, et comme détaillé ci-après, l'équipe UXFME était constituée de six développeurs, un Product Owner, un Scrum Master et un expert. Ce total de 8 personnes plus l'expert intervenant régulièrement apparaît donc être raisonnable.
L'équipe en charge du développement d'UXFME a déjà été par le passé de taille plus importante, jusqu'à atteindre une quinzaine de personnes. Pour rester efficace, l'équipe a donc été séparée en deux équipes distinctes avec deux Scrum Master, leur propre périmètre d'action, et donc leur propre backlog Scrum. La planification des drops et sprints était alors commune et le Product Owner, unique, participait régulièrement aux deux mêlées quotidiennes.

[8] BRANDT R., 2011, Jeff Bezos of Amazon: Birth of a Salesman, The Wall Street Journal

Organisation

En prenant en compte la multitude de composants de la suite logicielle UXFME, la pluridisciplinarité de l'équipe la développant apparaît nécessaire. Les membres de l'équipe sont expérimentés et pour la majorité vétérans du développement mobile et/ou serveur.

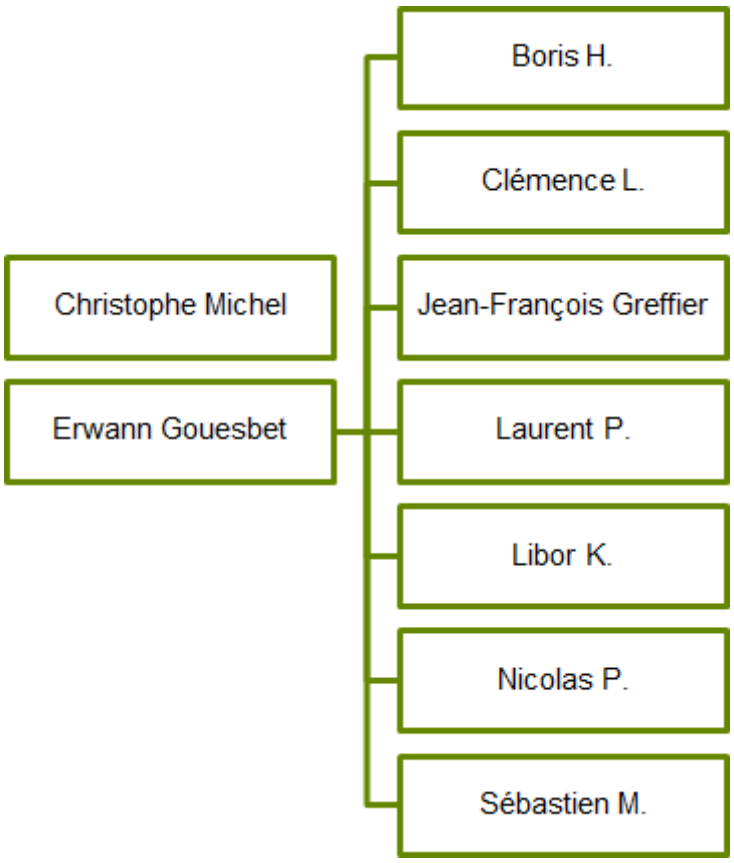


Figure 8 : Organigramme de l'équipe UXFME

Durant la période de l'étude décrite dans ce mémoire, l'équipe UXFME se constitue comme suit :

- Boris H. est ingénieur test et validation senior.
- Clémence L. agit en fin de drop en tant que rédacteur technique. Son action s'inscrit donc dans la notion d'expert de scrum.
- Jean-François GREFFIER, l'auteur. Développeur senior dans le mobile, avec des compétences web.
- Laurent P Développeur JEE
- Libor K. Développeur JEE, Libor est en télétravail depuis la République Tchèque.

- Nicolas P. est développeur senior dans l'embarqué puis le mobile.
- Sébastien M. Développeur JEE, expert infra.

L’équipe de management :

- Christophe MICHEL, Product Manager
    - Définit la stratégie du produit et sa feuille de route
    - Fait de la veille du marché et de la compétition
    - Promeut le produit (en interne et externe)
    - Fait l’interface avec les responsables produit des autres solutions de la suite Amdocs
- Erwann GOUESBET, manager de l’équipe et Program Manager
    - Gère les activités afin de répondre aux exigences du projet
    - Assure l’alignement des calendriers (synchronisation avec d'autres projets)
    - Facilite la formation professionnelle
    - Fait l’interface avec les responsables d’équipes des autres solutions de la suite Amdocs

C’est dans ce cadre, et de par mes compétences à la fois dans le domaine du développement des technologies web et mobiles, que j’ai naturellement été le contributeur principal de l’étude et de la réalisation d’un nouveau composant d’UXFME émulant le comportement d’un téléphone mobile.

3 Étude

Cette partie a pour but de décrire les différentes composantes de l'étude qui a été nécessaire au projet.
Au début de tout projet il est nécessaire de s'attacher à clarifier et préciser les besoins utilisateurs. Les contraintes techniques apportées par l'utilisateur et l'environnement d'UXFME seront également explicitées dans une première partie.
Ensuite seront exposés la méthodologie et les résultats d'une étude portant sur les solutions existantes pouvant assurer l'émulation d'applications mobiles hybrides.
Enfin la dernière partie détaille l'architecture logicielle proposée qui découle à la fois des besoins utilisateurs, des contraintes et des résultats de cette étude.

3.1 Problème posé : un émulateur pour UXFME

L'entité Amdocs en charge du produit UXF avait un projet d'environnement de développement intégré. L'IDE en devenir nommé UXFSuite devait permettre aux développeurs utilisant le User eXperience Framework de créer plus facilement et plus rapidement des applications s'intégrant aux produits back-end d'Amdocs. L'ambition était de créer un IDE s'exécutant depuis un navigateur web, voire de proposer un logiciel en tant que service.

Pour pouvoir offrir des outils de développement pour mobiles, l'équipe UXF a donc demandé à ce que UXFME s'intègre de deux façons avec UXFSuite. Tout d'abord en offrant la possibilité de configurer et packager des applications mobiles hybrides via un module Node.js implémentant une API documentée. Cette réalisation, le "Build Service", sort du cadre de ce document. Ensuite, en réalisant un émulateur d'applications hybrides mobiles s'intégrant dans l'IDE.

A cette étape préliminaire deux problématiques se présentent.
Premièrement, et même si certaines demandes sont claires, les besoins ne sont pas tous clairement explicités par l'équipe en charge d'UXF. Il faut donc les clarifier et même proposer une vision basée sur l'expérience acquise dans le domaine du mobile par l'équipe en charge d'UXFME. Par ailleurs, une étude permettra de dresser un état des lieux précis des solutions existantes.

Deuxièmement, un couplage trop fort avec une solution en particulier n'est en soit pas souhaitable. En effet, UXFME permet de créer des applications mobiles hybrides avec des solutions web autres que UXF. Par ailleurs, le développement de cet émulateur est une excellente opportunité d'enrichir les solutions déjà proposées par UXFME pour faciliter la création et le débogage d'applications mobiles hybrides, indépendamment d'un IDE en général, et de celui d'UXF en particulier.

3.1.1 Expression des besoins

Les souhaits exprimés par l'équipe UXF sont comme suit :

- Tout d'abord, et puisque UXFSuite l'est, l'émulateur UXFME doit être accessible depuis un navigateur Internet. S'il n'est pas forcément intégré fortement dans le logiciel UXFSuite, l'émulateur doit être idéalement accessible depuis le même environnement.
- Ensuite, la certification d'appareils UXFME "Device Certification" garantit un niveau de support contractuel et permet d'avoir des données précises et tirées de tests sur appareils mobiles. Les profils d'appareils accessibles depuis l'émulateur doivent donc être issus des données des Device Certification.
- Il est nécessaire de gérer l'ensemble des fonctionnalités clefs spécifiques aux appareils mobile et mises à disposition par la solution UXFME : les APIs client UXFME, la gestion du cycle de vie des applications, et la notification push.
- L'émulateur UXFME doit proposer des outils de débogage web, ou s'intégrer avec des outils existants. Ces outils doivent proposer un large panel de fonctionnalités, qui sont des standards d'usage pour les développeurs web : inspection DOM, réseau, accès aux sources, outils de profilages, console. Ce sont là les fonctionnalités classiques d'outils de débogage embarqués dans les navigateurs web.
- Enfin, l'émulateur doit être facile à déployer sur un poste de développeur. C'est-à-dire ne pas engendrer de contraintes fortes (pré-requis, procédure de déploiement…) et être compatible avec l'environnement technique.

Par ailleurs, il apparaît évident que les principales caractéristiques d'un appareil mobile doivent être émulées. Par exemple les caractéristiques physiques de l'affichage qui sont impérativement à prendre en compte par un développeur

d’application mobile, mais aussi les fonctionnalités HTML5 liées aux capteurs généralement présents sur un smartphone ou une tablette.
Néanmoins, l’émulateur n’a pas vocation à remplacer des tests sur un large panel d’appareils mobiles réels qui reste dans tous les cas nécessaires. La réalisation d'un émulateur d’applications hybrides mobiles a pour objet d'être une aide pour les développeurs.

3.1.2 Contraintes techniques et légales

Les contraintes découlent à la fois des utilisateurs et de l'environnement d’Amdocs.

Les utilisateurs potentiels de UXFSuite, et donc de l’émulateur, sont a priori les mêmes que ceux d’UXFME c’est-à-dire des développeurs web, le plus souvent employés d’Amdocs. Deux navigateurs sont en général supportés par les produits internes à Amdocs : Internet Explorer et Chrome. Il faut donc que l’émulateur supporte au moins un de ces deux navigateurs.

L’intégration et la modification de briques logicielles libres sont des pratiques courantes en développement informatique. Cela permet souvent de réduire le coût, d’accélérer le développement, mais aussi de garantir la qualité et la robustesse par l’utilisation de composants souvent éprouvés, gage de qualité. Cependant cette pratique apporte une contrainte légale.
En effet, si les logiciels libres et open-source (FOSS : Free and Open Source Software) sont gratuits, utilisables par tous (selon les conditions du logiciel), ouverts et modifiables, ils sont soumis à des licences d’utilisation pouvant être contraignantes.

Il existe trois types de licences libres et open-source :

- **Virale**. Elle ne peut pas être utilisée dans un produit commercial. En effet, ses conditions de publication du code source s’applique aux travaux dérivés. Exemple : GPL (GNU General Public License)
- **Non virale restrictive**. Les conditions de cette licence ne s’appliquent qu’au code du FOSS. Exemples : EPL (Eclipse Public License), CDDL (Common Development and Distribution License)

- **Non virale permissive**. Beaucoup plus souple, elle n'impose quasiment pas de conditions. Du code utilisant ce type de licence peut être intégré dans un logiciel libre, gratuit ou commercial. Exemples : BSD (Berkeley Software Distribution), Apache, licence MIT (du Massachusetts Institute of Technology)

Amdocs n'accepte pas l'intégration de code soumis à certaines licences, comme les licences de type viral, à cause des risques encourus du point de vue légal. Le premier risque est celui d'être poursuivi en justice pour non-respect d'une licence. Le second est la possibilité d'une injonction interdisant la distribution d'un produit Amdocs. Enfin dans le cas d'une licence virale, il est possible qu'une cour de justice oblige l'entreprise à dévoiler son code source ou fournisse son logiciel gratuitement.

L'intégration ou la modification de logiciels libres et open-source est donc soumis à une procédure interne à Amdocs, où certaines licences ne sont pas utilisables et où le code source tierce doit être déclaré afin d'être étudié avant acceptation. La partie réalisation de ce mémoire détaille cette procédure pour les FOSS introduits pendant le développement de l'émulateur UXFME.

3.1.3 Émulateur ou simulateur

L'émulateur est un logiciel qui présente le même comportement externe qu'un dispositif réel, qui est donc émulé. Il est donc possible de faire fonctionner sur l'émulateur un logiciel prévu pour l'appareil réel.

Par opposition, un simulateur est une application plus simple qui reproduit quelques comportements d'un système, sans toutefois de lien direct avec un appareil existant. Par exemple le simulateur iOS ne reproduit pas le comportement d'un téléphone iPhone mais celui d'un appareil mobile théorique. En effet il n'est pas possible de lancer directement un logiciel pour iPhone sur ce simulateur, qui n'utilise pas la même architecture de processeur. Le simulateur peut être considéré comme une cible de développement spécifique puisqu'il faut recompiler le logiciel.

Pour ce projet, on parle bien d'émulation dans le cadre d'application mobile hybride. Le but est d'aider les développeurs à mettre au point leur applications sans pour autant

reproduire dans le détail l'implémentation de HTML5 ou de JavaScript de l'appareil émulé, ou le niveau de support des APIs UXFME. Néanmoins, une application UXFME fonctionnera sans modifications dans l'émulateur.

3.2 Étude de solutions préexistantes

Un état de l'art a été effectué pour déterminer si des solutions du marché peuvent être réutilisées, même partiellement, et identifier dans ce cas ce qui doit être développé pour couvrir l'ensemble des besoins.
Dans le cadre de Scrum une étude technique de ce type prend la forme d'une story d'exploration spécifique : la spike.

3.2.1 Méthodologie de l'étude

La première phase de l'étude a consisté à réaliser une recherche extensive des outils existants permettant d'émuler une application mobile hybride ou un navigateur Internet mobile. Certains outils ont été exclus car incompatibles avec les contraintes techniques ou l'environnement de l'entreprise Amdocs. C'est par exemple le cas de solutions de cloud computing et/ou payantes comme par exemple Genymotion ou Bluestacks.
Les solutions suivantes ont été retenues pour l'étude :

- Les émulateurs natifs pour Android, iOS et Windows Phone
- Apache Ripple, un émulateur d'application mobile hybride
- Chrome DevTools et son mode d'émulation mobile

La seconde phase de l'étude porte sur l'évaluation de ces outils sous trois angles : fonctionnalités, possibilité d'étendre et d'enrichir la solution, et vérification du bon fonctionnement des fonctionnalités les plus importantes par des tests.

Les fonctionnalités attendues listées ci-après découlent de l'expression des besoins. La présence de ces fonctionnalités et le niveau de support de celles-ci par les solutions évaluées constituent les critères pour cette étude.

Tableau 5 : Fonctionnalités attendues

Fonctionnalité	Commentaires
Sélection d'appareil mobile	Les appareils issus de la Device Certification doivent être présents par défaut
Configuration d'appareil	En prenant en compte la définition et la taille physique de l'écran
Surcharge du User-Agent	Prédéfini avec des valeurs issues de la Device Certification
Rendu graphique	En prenant en compte la définition et la taille physique de l'écran
Géolocalisation	Latitude + longitude
Accéléromètre	x + y + z
Orientation	Alpha + beta + gamma
Limitation de bande passante	Bande passante et latence
Outils de développement web	Incluant inspection DOM, réseau, accès aux sources, outils de profilages, console

La possibilité d'étendre et d'enrichir une solution, en plus de la faisabilité technique, est dépendante d'une contrainte légale. En effet, la conformité du logiciel open-source que l'on veut modifier avec la politique FOSS d'Amdocs est indispensable. Il faut donc s'assurer à minima que la licence utilisée n'est pas proscrite par Amdocs.

Enfin, certaines fonctionnalités doivent être testées avec un contenu spécifique. Ces fonctionnalités et leurs tests sont détaillés ci-après. Ces tests sont importants car ils ont révélé que certaines fonctionnalités étaient incomplètes ou n'était pas fonctionnelles.

Surcharge du User-Agent

On désigne communément par “User-Agent” la chaîne renvoyée par une application cliente via l’entête HTTP “User-Agent”. Sur le web, un agent utilisateur permet par exemple d’identifier un navigateur, ou un robot d’indexation.

Dans le cadre d’un émulateur, la surcharge de la chaîne d’agent utilisateur permet de demander à un serveur des ressources destinées à la plateforme émulée, en se faisant passer de fait pour elle. Pour vérifier que la surcharge est effective, il faut accéder à l’information envoyée à un serveur distant.

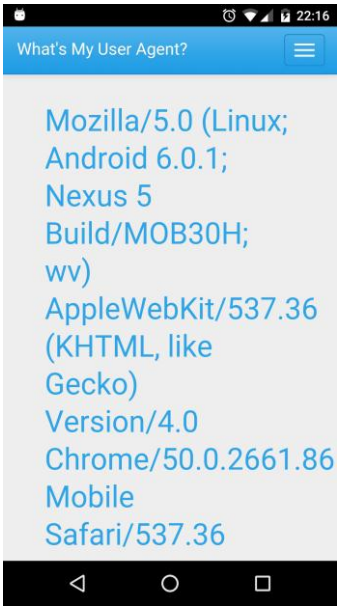


Figure 9 : Capture du site web whatsmyua montrant le User-Agent de Chrome sur un appareil Android

Par exemple en utilisant un service gratuit, on obtient sur un mobile :
Mozilla/5.0 (Linux; **Android 6.0.1**; **Nexus 5** Build/MOB30H) AppleWebKit/537.36 (KHTML, like Gecko) **Chrome/49.0.2623.105** Mobile Safari/537.36

On connaît donc ici le système d’exploitation (Android) et sa version, le modèle de smartphone (ici un Google LG Nexus 5), ainsi que le navigateur utilisé (Chrome) et sa version. Il apparaît donc que les informations d’agent utilisateur peuvent être très détaillées. La chaîne d’agent utilisateur est donc très importante car elle permet d’avoir des informations précises permettant d’adapter le contenu, ou de faire des statistiques d’utilisation.

Géolocalisation

La géolocalisation est le procédé qui permet de connaître la position géographique d'un objet. Dans le cas d'un appareil mobile, plusieurs procédés peuvent être utilisés :

- La triangulation par identifiant de cellule, la technique de géolocalisation mobile la plus courante, repose sur la position connue des antennes relais (GSM, UMTS, LTE). En identifiant les antennes relais disponibles, le mobile est capable d'estimer sa localisation. La précision va de 100m en milieu urbain à 10km en milieu rural.
- Le positionnement par satellite consiste à calculer la position grâce aux signaux de satellites. Le plus utilisé est le GPS (Global Positioning System). La précision est de 15m à 100m.
- Le A-GPS (Assisted-GPS) est une amélioration du positionnement par GPS. L'appareil mobile télécharge depuis Internet une table d'éphéméride des satellites, permettant d'accrocher le signal d'un satellite plus rapidement. La précision est la même que pour le GPS, mais la position est retrouvée plus rapidement.
- La géolocalisation depuis le contexte : l'adresse IP, un réseau WiFi connu, …

En pratique, un appareil mobile combine plusieurs de ces techniques pour connaître sa position.

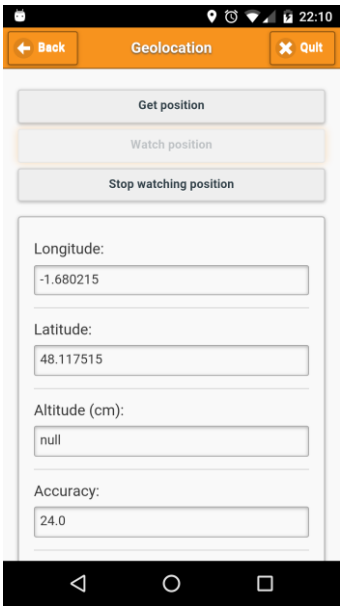


Figure 10 : Contenu de test utilisé pour la géolocalisation

Pour tester cette fonctionnalité, on utilise un contenu spécifique affichant les données accessibles depuis l'API HTML5.

Accéléromètre et orientation

La majorité des appareils mobiles (tablettes et smartphones) embarquent des capteurs permettant de connaître l'orientation de l'appareil. L'orientation est en général connue grâce à plusieurs accéléromètres et à un gyroscope.
L'accéléromètre est un capteur permettant de mesurer l'accélération linéaire. On peut en utiliser trois pour les trois axes.
Le gyroscope quant à lui est un capteur de position angulaire.

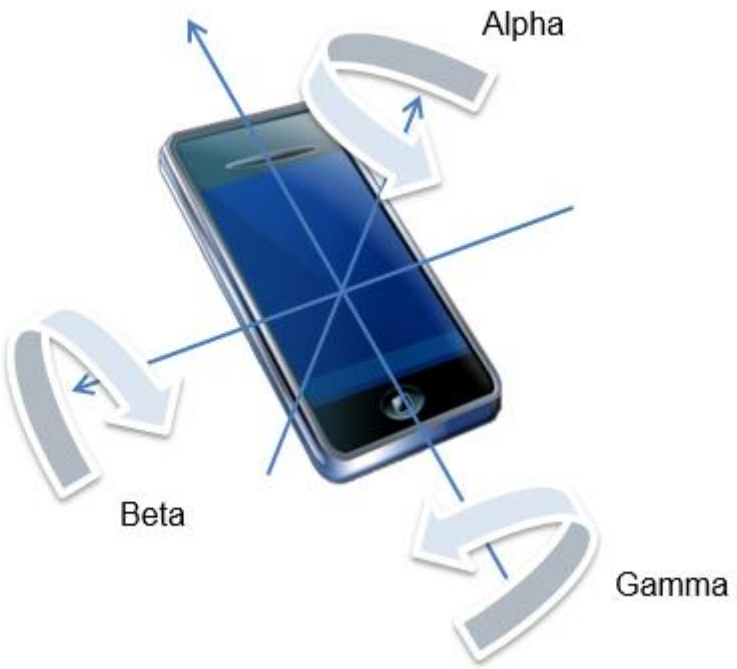


Figure 11 : Angles alpha, beta, gamma

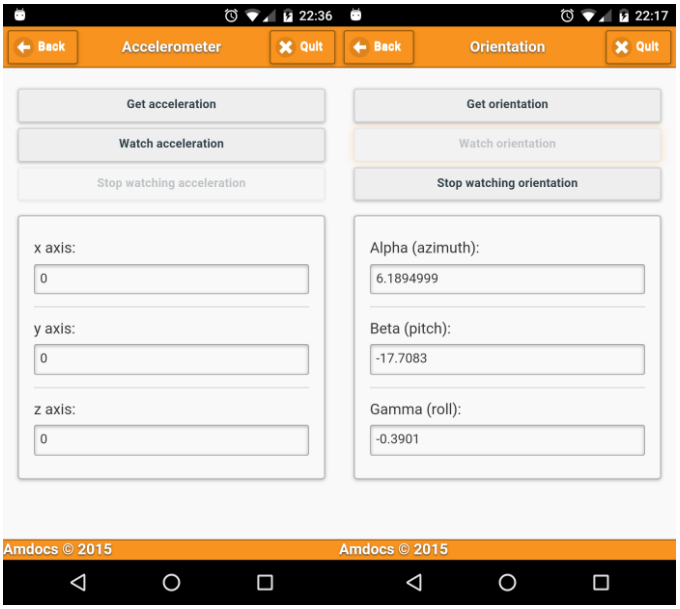


Figure 12 : Contenus de test utilisés pour l'accéléromètre et l'orientation

Pour tester cette fonctionnalité, on utilise un contenu spécifique affichant les données accessibles depuis l'API HTML5 : accélération linéaire sur les axes x, y et z ; accélération angulaire sur les angles alpha, beta et gamma.

Outils de développement web

Les outils de développement web permettant de déboguer en temps réel sont devenus indispensables aux développeurs. En effet seuls ces outils permettent d'inspecter et de tester en temps réel les différents aspects d'une application web.

Sous forme d'extensions, comme Firebug pour Firefox, ou intégrés au navigateur, comme les "Developer Tools" pour Chrome, ces outils sont tellement populaires que leur usage dépasse le périmètre du navigateur Internet seul. On le voit par exemple avec l'avènement du débogage web distant pour les applications web ou les applications mobiles hybrides, permettant de déboguer le navigateur ou la WebView d'un appareil mobile. Ces outils de développement sont aussi disponibles pour Node.js avec des modules comme node-inspector.

Les fonctions attendues d'un tel outil sont :

- inspection DOM
- réseau
- sources
- profilage mémoire et processeur
- console JavaScript

Le Document Object Model (DOM) est une interface qui permet d'accéder et de mettre à jour le contenu, la structure et le style d'un document HTML ou XML (Extensible Markup Language). Il est possible avec un outil de développement web de parcourir la structure du DOM pour inspecter un élément : son contenu, ses propriétés HTML ou le style associé. Le développeur peut alors mettre à jour ces propriétés et observer le changement directement dans le rendu du navigateur.
L'inspection DOM est un élément essentiel de débogage car il permet d'inspecter et de mettre à jour en temps réel des propriétés qui se reflètent le plus souvent dans le rendu de la page web, accélérant par exemple les tâches de mise en page ou de design.

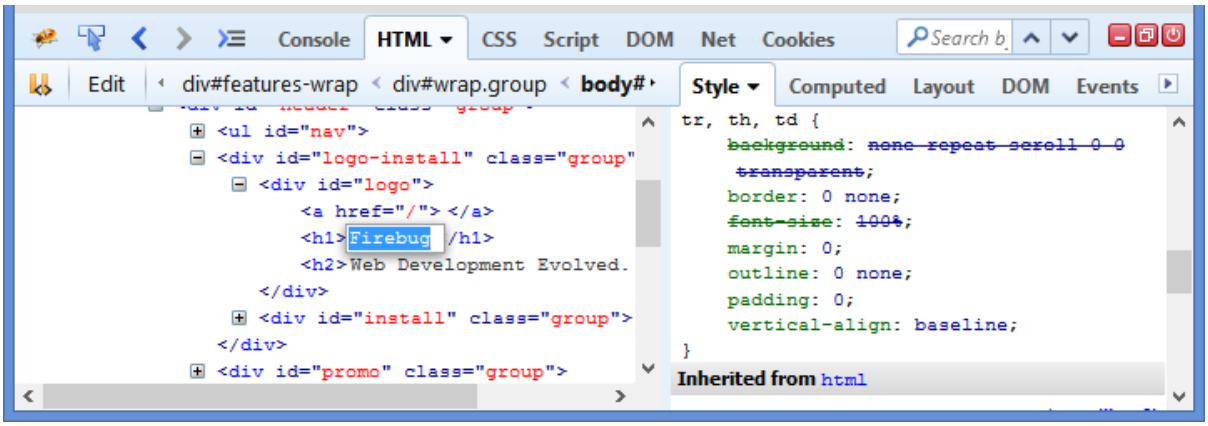
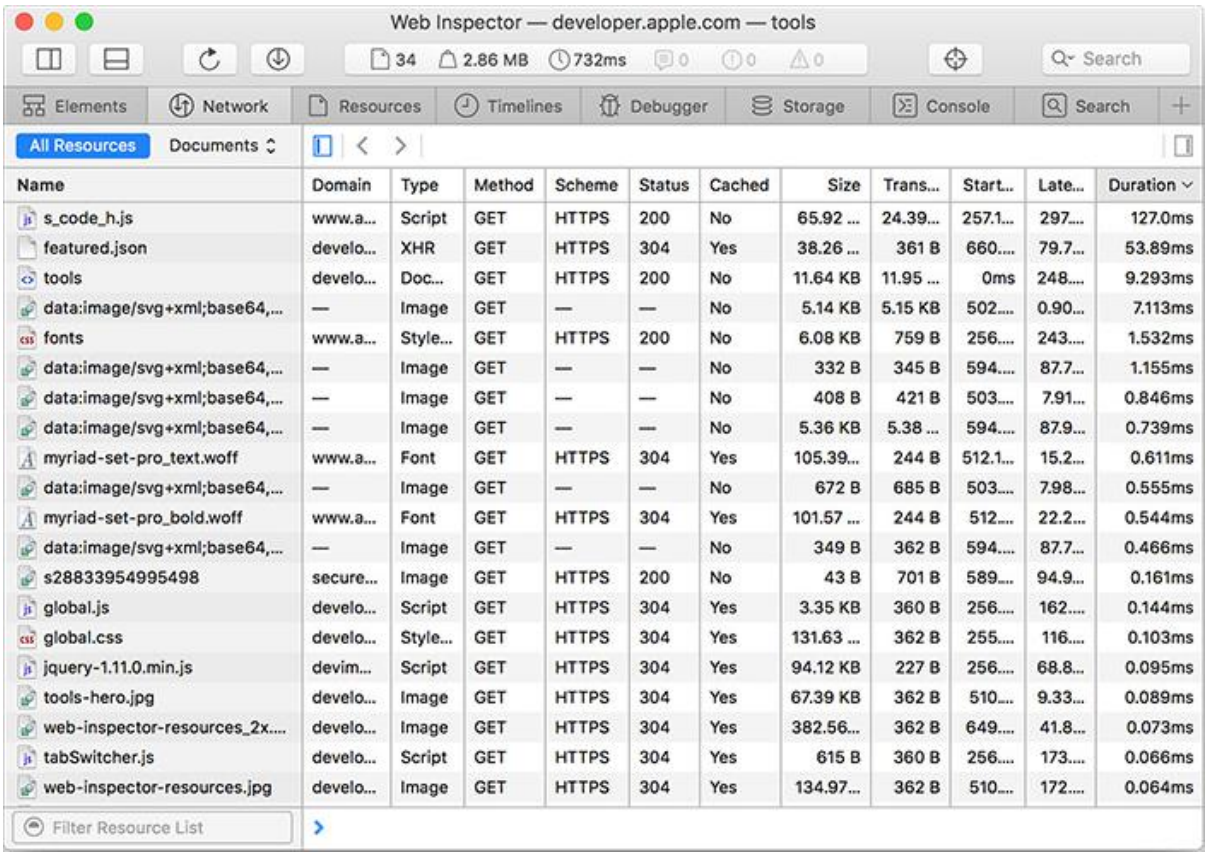


Figure 13 : Inspection DOM dans l'extension Firebug

Alors que les contenus web sont de plus en plus dynamiques, l'inspection des requêtes réseau est devenue indispensable. En effet, la majorité des applications web ou mobiles hybrides chargent du contenu dynamiquement : images, scripts, fontes… Les applications font aussi appel à des serveurs via des services web.

La vue d'inspection permet d'observer l'enchaînement des requêtes réseau, d'analyser la réponse du serveur, ou de rejouer des requêtes. Il est alors plus aisé de déboguer une application, ou d'optimiser ses performances après analyse.



Name	Domain	Type	Method	Scheme	Status	Cached	Size	Trans...	Start...	Late...	Duration
s_code_h.js	www.a...	Script	GET	HTTPS	200	No	65.92 ...	24.39...	257.1...	297....	127.0ms
featured.json	develo...	XHR	GET	HTTPS	304	Yes	38.26 ...	361 B	660....	79.7...	53.89ms
tools	develo...	Doc...	GET	HTTPS	200	No	11.64 KB	11.95 ...	0ms	248....	9.293ms
...	—	Image	GET	—	—	No	5.14 KB	5.15 KB	502....	0.90...	7.113ms
fonts	www.a...	Style...	GET	HTTPS	200	No	6.08 KB	759 B	256....	243....	1.532ms
...	—	Image	GET	—	—	No	332 B	345 B	594....	87.7...	1.155ms
...	—	Image	GET	—	—	No	408 B	421 B	503....	7.91...	0.846ms
...	—	Image	GET	—	—	No	5.36 KB	5.38 ...	594....	87.9...	0.739ms
myriad-set-pro_text.woff	www.a...	Font	GET	HTTPS	304	Yes	105.39...	244 B	512.1...	15.2...	0.611ms
...	—	Image	GET	—	—	No	672 B	685 B	503....	7.98...	0.555ms
myriad-set-pro_bold.woff	www.a...	Font	GET	HTTPS	304	Yes	101.57 ...	244 B	512....	22.2...	0.544ms
...	—	Image	GET	—	—	No	349 B	362 B	594....	87.7...	0.466ms
s28833954995498	secure...	Image	GET	HTTPS	200	No	43 B	701 B	589....	94.9...	0.161ms
global.js	develo...	Script	GET	HTTPS	304	Yes	3.35 KB	360 B	256....	162....	0.144ms
global.css	develo...	Style...	GET	HTTPS	304	Yes	131.63 ...	362 B	255....	116....	0.103ms
jquery-1.11.0.min.js	devim...	Script	GET	HTTPS	304	Yes	94.12 KB	227 B	256....	68.8...	0.095ms
tools-hero.jpg	develo...	Image	GET	HTTPS	304	Yes	67.39 KB	362 B	510....	9.33...	0.089ms
web-inspector-resources_2x....	develo...	Image	GET	HTTPS	304	Yes	382.56...	362 B	649....	41.8...	0.073ms
tabSwitcher.js	develo...	Script	GET	HTTPS	304	Yes	615 B	360 B	256....	173....	0.066ms
web-inspector-resources.jpg	develo...	Image	GET	HTTPS	304	Yes	134.97...	362 B	510....	172....	0.064ms

Figure 14 : WebInspector, l'outil intégré à Apple Safari

A la manière d'un IDE, il est possible d'accéder à l'ensemble des ressources de la page web (HTML, JavaScript, CSS, images, vidéos...) mais aussi de mettre un point d'arrêt dans un source JavaScript. DevTools, l'outil de développement intégré à Google Chrome propose même aux développeurs d'éditer des fichiers locaux.

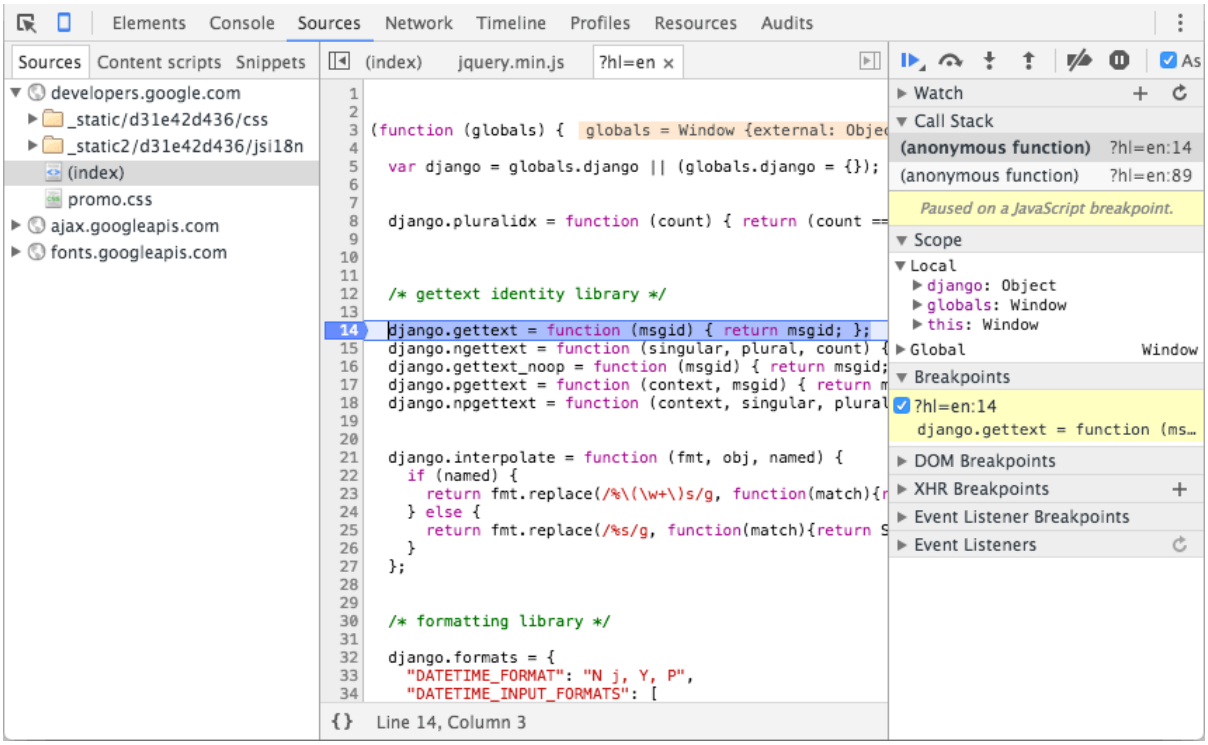


Figure 15 : Sources dans Chrome

Pour optimiser une application web, ou trouver la source d'un problème mémoire (fuite mémoire...), il est possible de profiler une application web sur deux critères. Cet outil extrêmement avancé permet de profiler l'utilisation du processeur ou de l'utilisation de la mémoire.

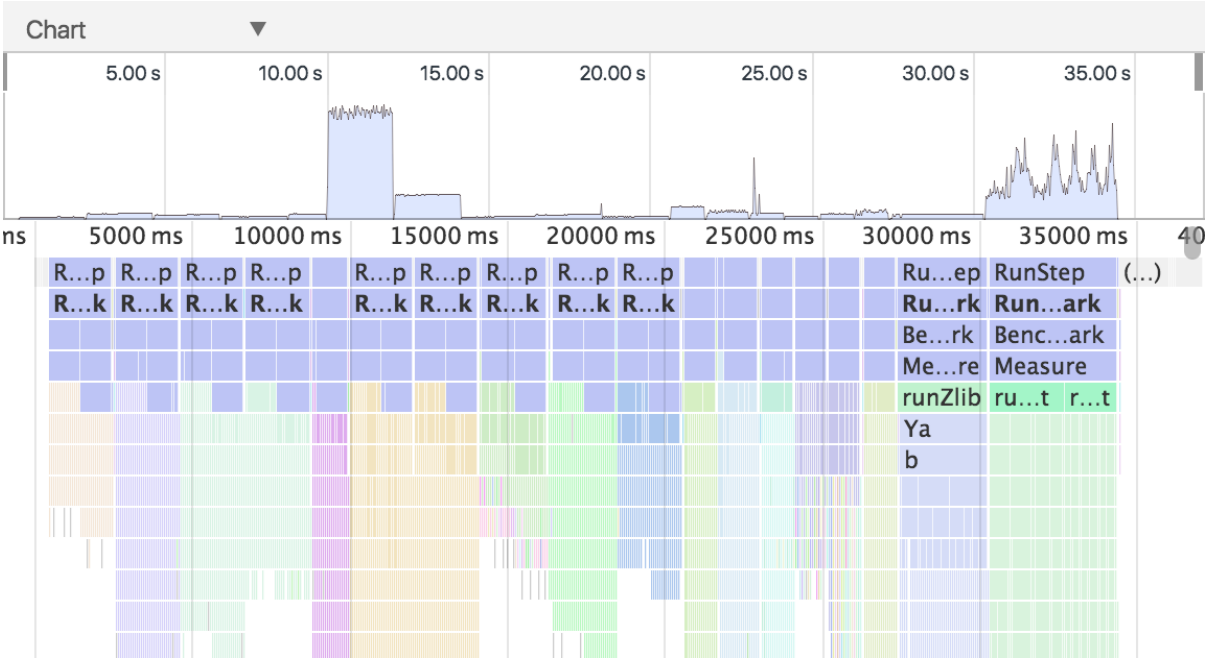


Figure 16 : Profilage dans Google Chrome

Enfin, la console JavaScript permet de récupérer les logs fournis par l'API JavaScript console.log(), mais aussi des détails sur les exceptions. De plus, cette console permet d'exécuter du code. Cela permet de tester extrêmement rapidement du code, par exemple dans une phase de prototypage.

3.2.2 Émulateurs natifs

Les émulateurs proposés par les constructeurs ou les éditeurs de système d'exploitation mobile sont très proches d'un appareil mobile réel. Ils peuvent donc être utilisés pour émuler une application mobile hybride ou une application web.

Depuis toujours les kits de développement pour appareils mobiles ont dû proposer des émulateurs aux développeurs. En effet, cette solution permet de tester une application plus rapidement sans avoir à déployer sur un appareil réel. Elle permet également d'émuler un appareil que l'on n'a pas à disposition. Ces logiciels prennent

en général l'émulation de toute la partie logicielle dont le système d'exploitation, mais également de la partie matérielle.

Figure 17 : Emulateur Android

Installation

Dans le cas d'UXFME il faut utiliser les émulateurs provenant respectivement du SDK Android, de Xcode pour iOS et de Visual Studio pour les plateformes Windows.

Ces émulateurs dit natifs suivent un même principe : une image système similaire à celle d'un véritable appareil mobile est lancée sur une configuration matérielle émulée. Pour atteindre ce résultat, il faut donc également émuler le matériel. En effet la majorité des appareils mobiles utilisent des processeurs d'architecture ARM, à opposer aux architectures x86 ou x64 généralement présentes sur ordinateurs personnels (PC ou Mac). De plus, il faut également émuler les différents capteurs de l'appareil mobile.
Un émulateur natif est donc très proche d'un appareil physique.

Sur Android, les images systèmes de l'émulateur fournies dans le kit de développement ne sont toutefois pas représentatives de la réalité des appareils mobiles. Il s'agit de la version AOSP (Android Open Source Project) c'est-à-dire la version open-source d'Android et non la version embarquant les services essentiels de Google ou les personnalisations des différents constructeurs. Or la WebView et Google Chrome ne sont pas présents par défaut dans AOSP.
Le simulateur iOS propose par opposition une expérience plus simple puisque l'ensemble des terminaux Apple, iPhone et iPad, sont utilisables.
L'émulateur de Visual Studio permet également de cibler l'ensemble des configurations des appareils Windows Phone.

De plus, certains environnements de développement imposent de fortes contraintes. Si le SDK Android est disponible pour Windows, Linux et MacOS, il est par contre indispensable d'avoir un Apple Macintosh pour iOS. Pour Windows Phone 8.1 un ordinateur avec au minimum Windows 8.1 est requis, ce qui n'est pas une configuration proposée par le service informatique d'Amdocs.

Fonctionnalités

La sélection des appareils à émuler est complexe : il faut dans certains cas créer une image système pour chaque appareil mobile. Par ailleurs, le démarrage de l'émulation est relativement long et il n'est pas possible de changer d'appareil à la volée.
Selon les plateformes, il n'est pas possible de limiter la bande passante.
Enfin, les outils de développement web sont les même que sur un appareil réel c'est-à-dire limités aux solutions de débogage à distance. Ils sont absents sur certaines plateformes Windows et plus généralement difficile à mettre en place.

Possibilité d'extension

L'émulateur Android est open-source, ce qui n'est pas le cas des émulateurs iOS et Windows Phone. Néanmoins, modifier une image système d'Android revient à modifier l'ensemble du système d'exploitation ou une partie de ses composants systèmes (comme la WebView). Ces modifications de par leur complexité et leur coût ne sont pas possibles dans le cadre de projet d'émulateur UXFME.

Tests

Les tests portant sur la surcharge du User-Agent, de la géolocalisation, de l'accéléromètre et l'orientation n'ont révélé aucuns dysfonctionnements.

Conclusion concernant les émulateurs natifs

Des fonctionnalités importantes tel que la sélection d'appareil mobile ou les outils de développement web ne sont pas utilisables en l'état. Par ailleurs, les émulateurs natifs imposent des contraintes techniques et matérielles qui ne sont pas compatibles avec l'environnement typique d'un développeur web travaillant à Amdocs.
Ces outils sont avant tout destinés aux développeurs d'applications mobiles natives et ne semblent donc pas adaptés à des utilisateurs finaux ou à des développeurs web.

3.2.3 Apache Ripple

Apache Ripple est un émulateur pour applications mobiles HTML5 et les frameworks d'application mobiles hybrides Apache Cordova et BlackBerry WebWorks.

Historiquement développé par BlackBerry, Ripple est maintenant un projet open-source de la fondation Apache, principalement utilisé avec Apache Cordova. Apache Ripple a été développé avec des technologies web (HTML5/JS/CSS), il fonctionne donc via un navigateur web. Son but est de faciliter le développement et le test d'applications web mobiles et d'applications hybrides mobiles. De par sa nature web et son architecture, Apache Ripple permet d'utiliser des outils d'inspection et de débogage standards du monde du développement web.

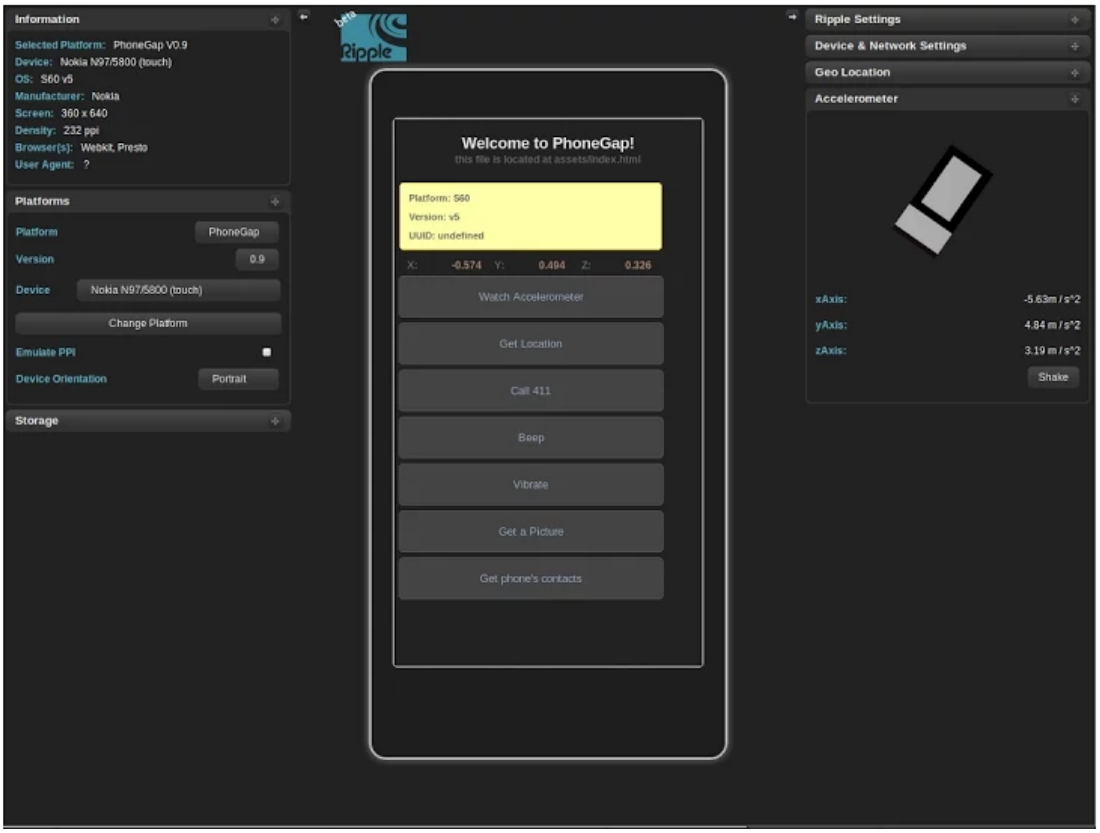


Figure 18 : Capture d'écran de Apache Ripple

Installation

L'émulateur Apache Ripple est disponible sous deux formes. Tout d'abord un module Node.js qui fait office de serveur web, ensuite une extension du navigateur Google Chrome.

Node.js est un environnement d'exécution pour serveur s'appuyant sur le moteur JavaScript V8 de Google Chrome. Son approche par événements, la gestion asynchrone des entrées-sorties et son support du JavaScript en font une plateforme de plus en plus populaire.
Le module Node.js de Ripple est disponible en version 0.9.36 (à jour) sur le gestionnaire de paquet officiel de Node.js, NPM (Node Package Manager).

Disponible depuis 2008, Google Chrome s'est imposé comme navigateur Internet le plus utilisé. Son système d'extensions et d'applications permet d'ajouter de nombreuses fonctionnalités au navigateur. L'extension Ripple est disponible sur le magasin d'applications Chrome Web Store, en version 0.9.15. Pour avoir accès à une

version plus à jour il faut construire l'application depuis les sources.

Pour tester l'ensemble des fonctionnalités les prérequis sont donc :

- Node.js
- NPM
- Google Chrome

Par ailleurs les outils suivants sont nécessaires pour compiler Apache Ripple depuis le code source :

- Python
- Une chaîne de build C/C++ comme GCC

Fonctionnalités

La sélection des appareils à émuler se fait depuis un menu déroulant. Les appareils présents dans cette liste sont parmi les smartphones et tablettes les plus populaires. Par ailleurs cette liste est facilement modifiable dans le code source via un fichier de configuration. La limitation de la bande passante n'est pas présente. Les autres fonctionnalités attendues sont présentes.

Les outils de développement web sont fournis par le navigateur web dans lequel l'émulateur est exécuté. En effet Apache Ripple, en tant qu'extension ou de module Node.js, est développé en HTML5/JS/CSS. Il est donc possible de le déboguer comme une page web classique. Avec Chrome on peut donc inspecter le DOM des pages web émulées, analyser les transactions réseau, accéder aux sources, faire du profilage ou accéder à la console JavaScript.

Possibilité d'extension

Ripple, comme les autres produits de la fondation Apache, est distribué sous licence Apache. Cette licence permet la modification de code sous toute ses formes : libre ou propriétaire, gratuit ou commercial. Cette licence est par ailleurs compatible avec la politique FOSS d'Amdocs, puisque déjà étudiée et approuvée par les services juridiques de l'entreprise.
La solution est facilement modifiable, bien que manquant de documentation que ce soit à destination de l'utilisateur final ou du développeur voulant modifier le produit.

Apache Ripple est donc satisfaisant quant à la possibilité d'étendre et d'enrichir la solution.

Tests

Deux fonctionnalités testées ne sont pas opérationnelles : la surcharge du User-Agent et le rendu graphique. En effet, et malgré la présence annoncée de la fonctionnalité, la chaîne renvoyée via le header HTTP n'est pas surchargée et reste celle de la plateforme hôte de l'émulateur. Donc ici la valeur renvoyée par Chrome Windows.
Le rendu graphique n'est lui que partiellement émulé. En effet, le rendu graphique prend bien en compte la définition de l'appareil émulé mais pas la taille physique de l'écran.
Après test, il s'est révélé que le module Node.js et l'extension Chrome offrent le même niveau de fonctionnalités.

Conclusion concernant Apache Ripple

La plupart des fonctionnalités requises pour l'émulateur UXFME sont présentes. Avoir une solution développée avec des technologies web est ici un avantage, car elle permet aux développeurs web d'utiliser les outils habituels de débogage du navigateur. Néanmoins, certaines fonctionnalités critiques se sont révélées non fonctionnelles. Par ailleurs, le projet est très peu actif et manque de documentation.

3.2.4 Chrome DevTools

Les outils de développement de Google Chrome proposent un mode pour l'émulation de comportement web mobile.

Disponible depuis 2008 sur ordinateurs et 2012 sur appareils mobiles, Google Chrome s'est imposé comme le navigateur Internet le plus utilisé. Les outils de développement intégrés "DevTools" sont très complets et leur utilisation largement répandue. Le trafic mobile étant en nette progression, Google a décidé de proposer un mode permettant d'émuler le rendu web d'appareils mobiles. Au moment de l'étude, ce nouveau mode est à l'état de pré-version.

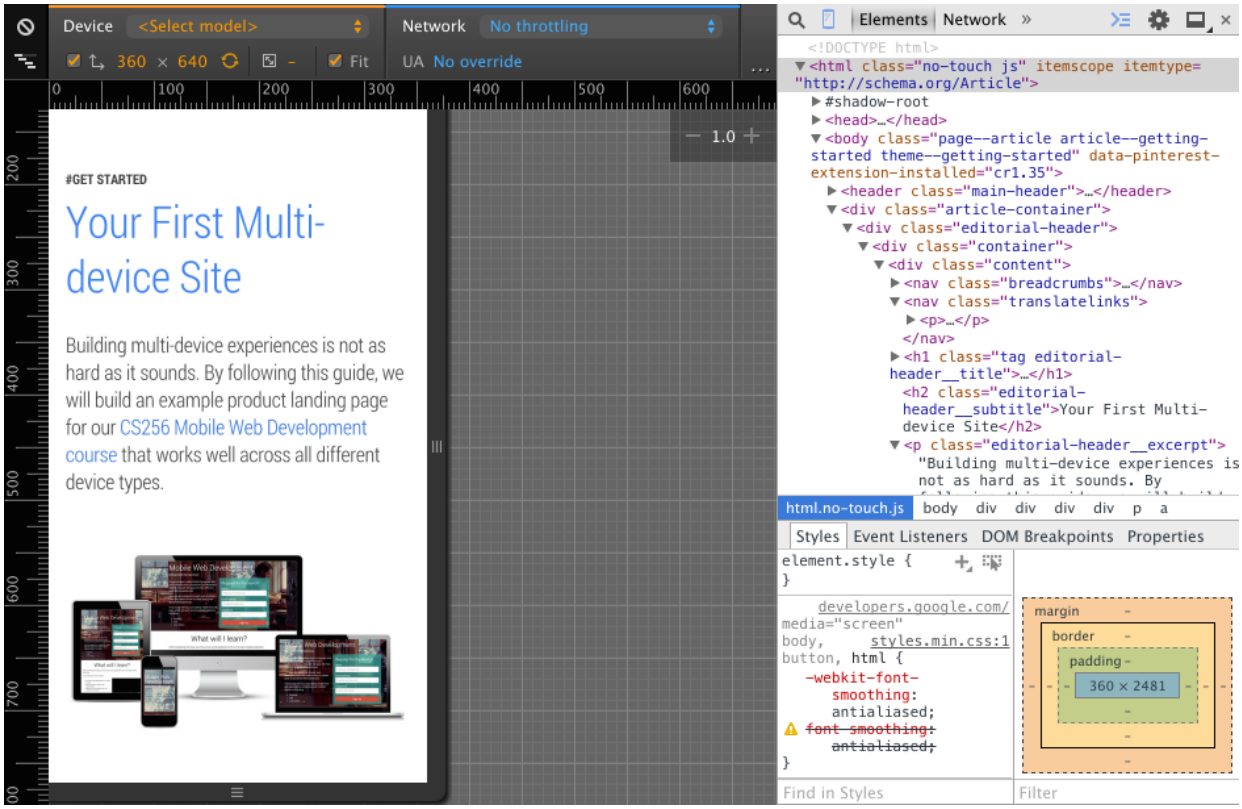


Figure 19 : Le mode device de Chrome Canary

Installation

Différentes versions de Google Chrome existent :

- La version stable est destinée aux utilisateurs finaux.
- La version Bêta est une pré-version de test de la version stable.
- La version de développement est la livraison de la fin de chaque cycle de développement. Cette version est testée mais est sujette aux bogues.
- Google Chrome Canary est conçu pour les développeurs et les premiers utilisateurs, c'est une version Alpha. Cette version reçoit des modifications et de nouvelles fonctionnalités, parfois expérimentales, de façon quasi quotidienne.

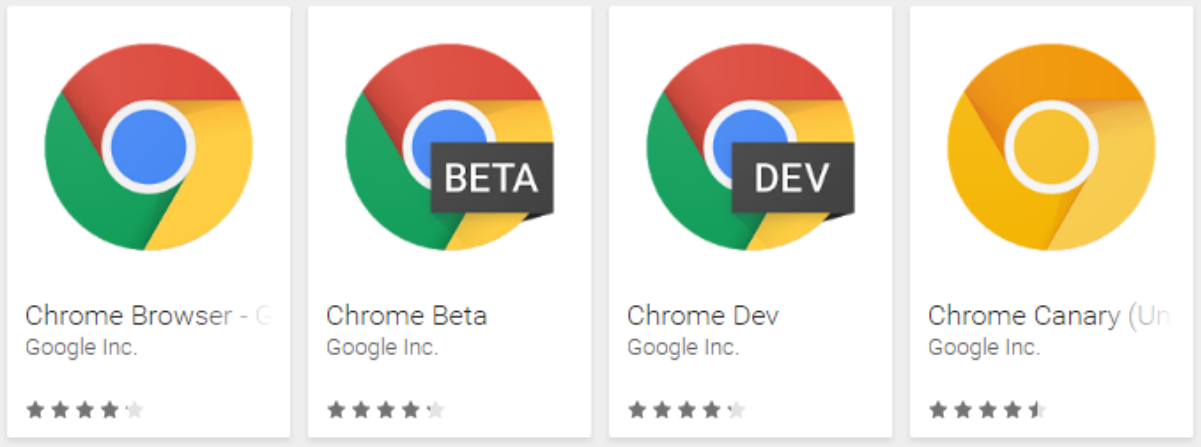


Figure 20 : Les différentes versions de Chrome

C'est cette version Canary qui a été utilisée pendant l'étude puisqu'elle était la seule qui embarquait le nouveau mode device des DevTools. Contrairement aux autres versions, il est possible d'installer Chrome Canary et la version stable de Google Chrome simultanément.

Google Chrome est déjà recommandé pour certains usages au sein d'Amdocs, comme le développement web.

Fonctionnalités

La sélection des appareils à émuler se fait depuis un menu déroulant préconfiguré. Il est possible, manuellement, d'ajouter des appareils mobiles à cette liste. Mais pour satisfaire aux exigences, il faut que cette liste puisse être préremplie avec les appareils issus de la Device Certification. Pour cela, il faut modifier les outils de développement de Chrome. En effet, après une étude approfondie des APIs disponibles, il est apparu que les autres modes d'extension n'étaient pas applicables.

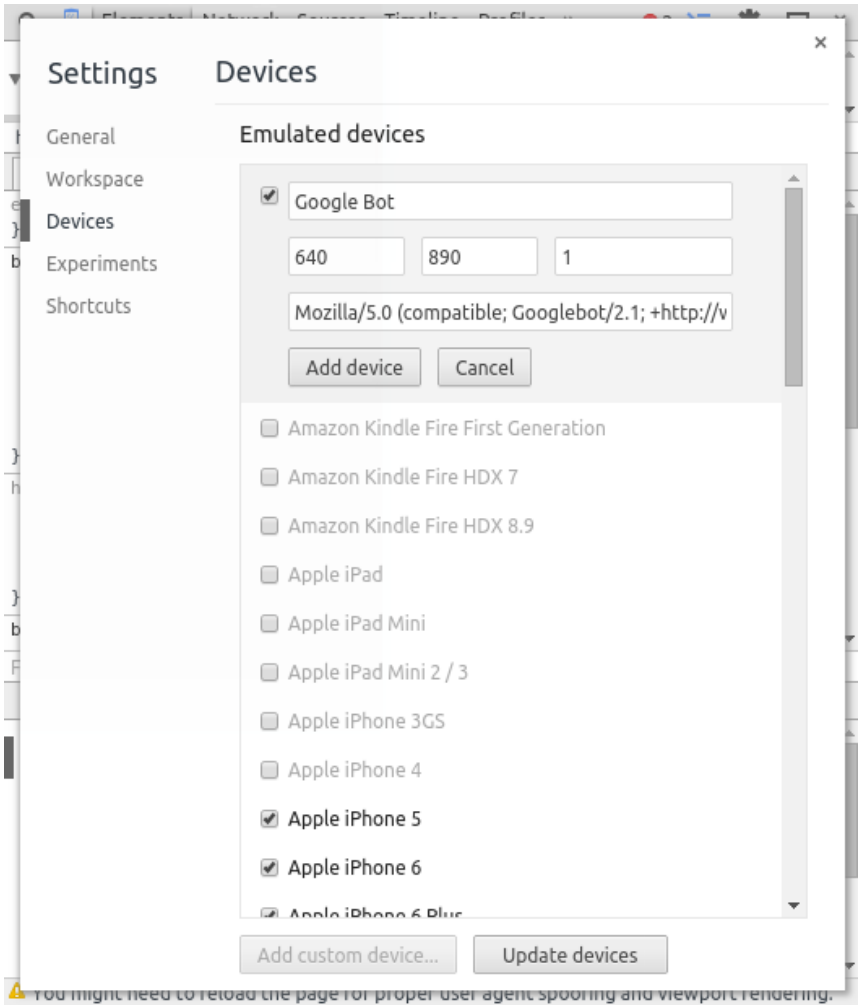


Figure 21 : Ajout d'une configuration à la liste d'appareils émulés

Possibilité d'extension

Google Chrome est basé sur le navigateur open-source Chromium. La plupart des composants sont distribués sous licence BSD, compatible avec la politique FOSS d'Amdocs.

Au moment de l'étude, il n'est pas possible de modifier les outils de développement DevTools sans modifier et recompiler Chromium ou faire une application indépendante. Chromium est un produit complexe qui requiert une chaîne de build sur une machine dédiée très puissante.

Tests

Après tests, la surcharge du User-Agent, de la géolocalisation, de l'accéléromètre et l'orientation apparaissent tous fonctionnels.

Conclusion concernant les Chrome DevTools

La majorité des fonctionnalités requises sont présentes, à part la sélection d'appareils venant de la Device Certification. L'absence de cette possibilité essentielle et l'impossibilité pratique de modifier Google Chrome écarte cette solution. De plus, le statut expérimental de la vue mobile de DevTools est problématique.

3.2.5 Synthèse de l'étude

A la fin de l'étude, il apparaît que les outils de développement de Chrome et sa vue mobile offrent la majorité des fonctionnalités demandées. C'est par ailleurs un outil très complet et un standard *de facto* dans l'industrie du développement logiciel web. Néanmoins, le choix ne s'est pas porté sur cette solution car l'outil en l'état est extrêmement difficile à modifier.

Apache Ripple, plus facile à modifier, ne remplit pas les conditions d'acceptation en l'état. Par ailleurs, l'avenir du projet est incertain et le risque de se baser sur un projet open-source peu actif et peu maintenu est grand.

Les émulateurs natifs ont été écartés car les différents environnements de développement imposent de fortes contraintes. De plus, il est virtuellement impossible de modifier ces solutions.

Tableau 6 : Synthèse de l'étude comparative des solutions préexistantes

	Natif	Apache Ripple	Chrome DevTools
Sélection d'appareil mobile	✗	✓	✗ Liste préconfigurée
Configuration d'appareil	✗	✓	✓
Surcharge du User-Agent	✓	✗ Non fonctionnel	✓
Rendu graphique	✓	✗ Non fonctionnel	✓
Géolocalisation	✓	✓	✓
Accéléromètre	✓	✓	✓
Orientation	✓	✓	✓
Limitation de bande passante	✗	✗	✓
Outils de développement web	✗	✓	✓
Licence	Multiples	Apache 2	BSD
Facilité de modification	✗	✓	✗

Légende

✓ supporté et fonctionnel
✗ non supporté ou non fonctionnel

Aucun des produits évalués ne répond suffisamment aux attentes pour être utilisé tel quel. La modification n'est pas non plus envisageable.

Il a donc été décidé de concevoir et d'implémenter entièrement l'émulateur pour le framework UXFME.

Bien que non-retenu, le produit Apache Ripple est intéressant de par son approche et son mode de fonctionnement. Apache Ripple sera donc une source d'inspiration forte pour la suite, notamment pour l'utilisation des outils de développement du navigateur comme l'inspection DOM.

3.3 Conception

Bien qu'aucun des produits détaillés dans l'étude des solutions préexistantes ne soit complètement adapté aux besoins exprimés, la conception et la réalisation de l'émulateur UXFME pourra mettre en avant certaines bonnes pratiques de ces solutions.

En partant des besoins du client, il est possible d'en déduire les fonctionnalités nécessaires et ainsi préparer les stories Scrum qui seront réalisées pendant les sprints de développement.

Pour chaque fonctionnalité seront listés les composants logiques importants menant ainsi à une architecture logicielle.

Enfin, l'ergonomie de l'application sera aussi détaillée avec un premier concept d'interface utilisateur.

3.3.1 Du besoin à la story

Pendant la phase d'expression des besoins et des contraintes du projet, des besoins non-fonctionnels ont été listés : accessibilité par le navigateur Internet, facilité de déploiement, compatibilité avec un des deux navigateurs recommandés dans l'entreprise (Internet Explorer et Chrome), respect de la politique FOSS d'Amdocs.

Quant aux besoins fonctionnels, ils permettent de lister les fonctions à réaliser. Selon l'importance de la fonctionnalité, elle peut être découpée en une ou plusieurs stories. La story étant un élément apportant de la valeur aux parties prenantes, elle peut prendre la forme d'une fonctionnalité ou d'un petit morceau de fonctionnalité. Ce sont ces récits utilisateurs qui seront réalisés pendant les différents sprints de développement.

Tableau 7 : Besoins et fonctionnalités correspondantes

Besoin	Fonctionnalité
Les profils d'appareils sont issus de la device certification	Sélection d'appareil mobile
Gérer l'ensemble des APIs client UXFME	Émulation des APIs UXFME
Gérer le cycle de vie des applications (mise à jour)	Émulation de la mise à jour d'application
Gérer les notifications push	Émulation de la notification push
Permettre le débogage web	Outils de développement web
Émuler l'affichage d'un appareil mobile	Rendu graphique Configuration du rendu graphique
Émuler les capteurs présents sur un appareil mobile	Émulation des APIs HTML5

Les stories permettant la création des fonctionnalités sont comme suit. A noter que la fonctionnalité "Outils de développement web" n'a pas de story dédiée. En effet, et suite à l'étude des solutions préexistantes d'émulateur pour application mobile hybride, il apparaît plus judicieux d'utiliser les outils de développement web fournis par le navigateur, offrant les fonctionnalités usuelles attendues par les développeurs.

Sélection d'appareil mobile

Il doit être possible de sélectionner un appareil parmi la liste des appareils de la UXFME device certification pour mettre à jour l'appareil émulé sur ces critères :

- User-Agent
- affichage incluant la définition et les caractéristiques physiques de l'écran
- APIs supportées ou non

Configuration des données de l'appareil émulé

Il doit être possible de configurer les données accessibles par l'émulation :

- liste des contacts
- liste des événements dans le calendrier
- système de fichiers

Configuration du rendu graphique

Il doit être possible de configurer l'affichage de l'appareil émulé :

- orientation de l'écran (portrait, paysage)
- barre de statut et ses différents modes

Émulation des APIs

Il doit être possible d'émuler l'ensemble des APIs client d'UXFME, ainsi que les APIs HTML5 géolocalisation, accéléromètre, orientation.

Mise à jour d'application

Il doit être possible de gérer le cycle de vie de l'application émulée, en spécifiant une instance du UXFME Applications Manager qui sera la source de mises à jour, afin de pouvoir en tester l'effet.

Émulation de la notification push

Il doit être possible depuis l'émulateur UXFME de forger et envoyer une notification push localement, qui agira sur la logique de l'application émulée comme une véritable notification.

3.3.2 Planning

La conception et le développement initial décrits dans ce mémoire se sont déroulés sur trois drops pour un total de 28 semaines. Chaque drop est constitué d'un ou plusieurs sprints de développement et d'un sprint de stabilisation.

Les sprints de développement sont dédiés à la conception, à l'architecture, à l'exploration technique ou au codage de produits. Plus généralement les sprints de développement sont dédiés à la réalisation de stories.

Le sprint de stabilisation est quant à lui dédié aux tests, à la correction de bogues mais aussi aux activités de livraison comme l'écriture et la mise à jour de la documentation ou le packaging des différents produits constituant la suite UXFME.

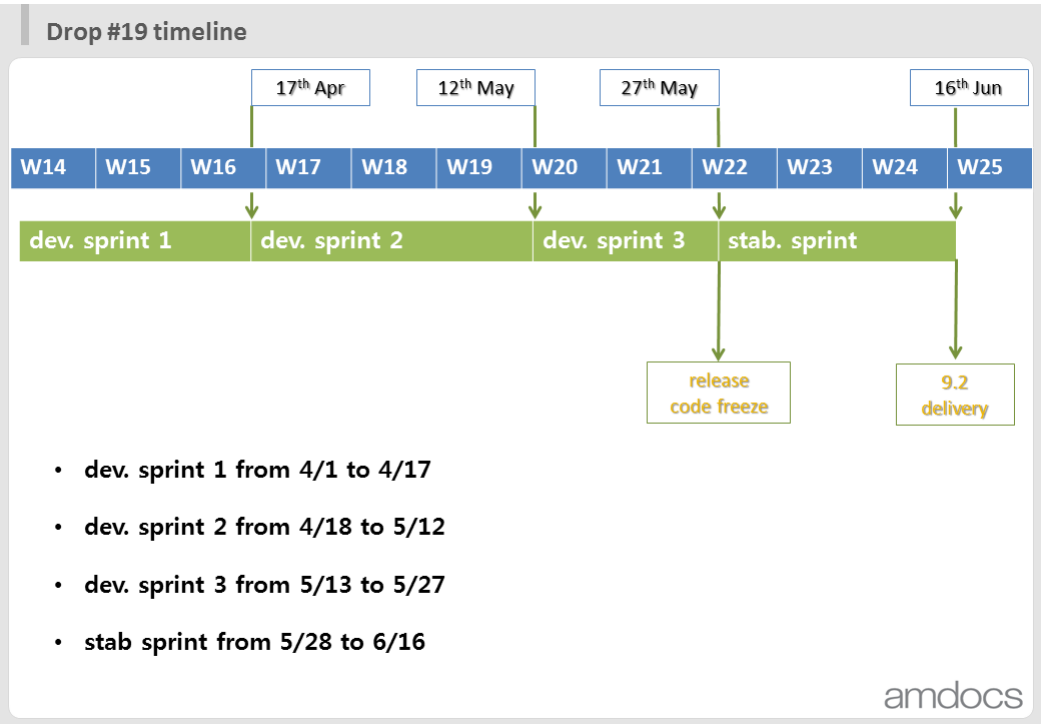


Figure 22 : Drop 19

Initialement prévu jusqu'au 16 juin, le drop 19 s'est exceptionnellement prolongé jusqu'au 30 juin. En effet, il a fallu livrer un lot de corrections pour ce drop sur des fonctions importantes liées à HTTPS sur le client UXFME, des modifications concernant le respect des contraintes de la suite Amdocs CES 9.2 et d'autres bogues clients… La phase de stabilisation a donc couru deux semaines de plus que la date initialement prévue.

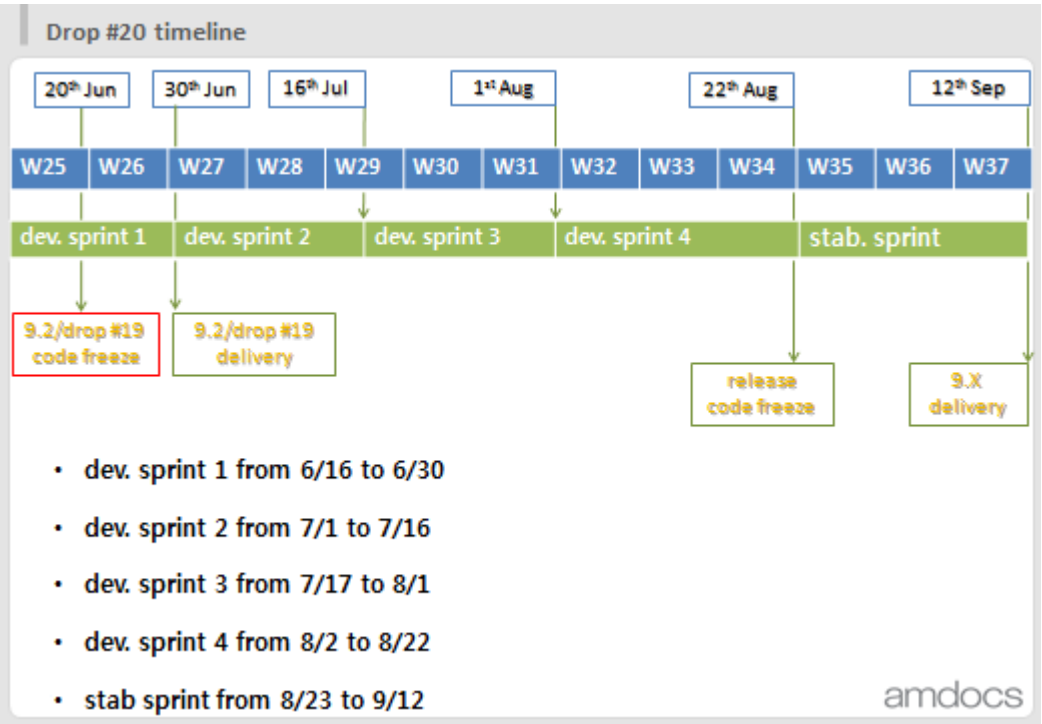


Figure 23 : Drop 20

Le drop 20 a démarré en parallèle pour une partie de l'équipe. La réalisation de l'émulateur décrite dans ce document a eu lieu durant cette période.

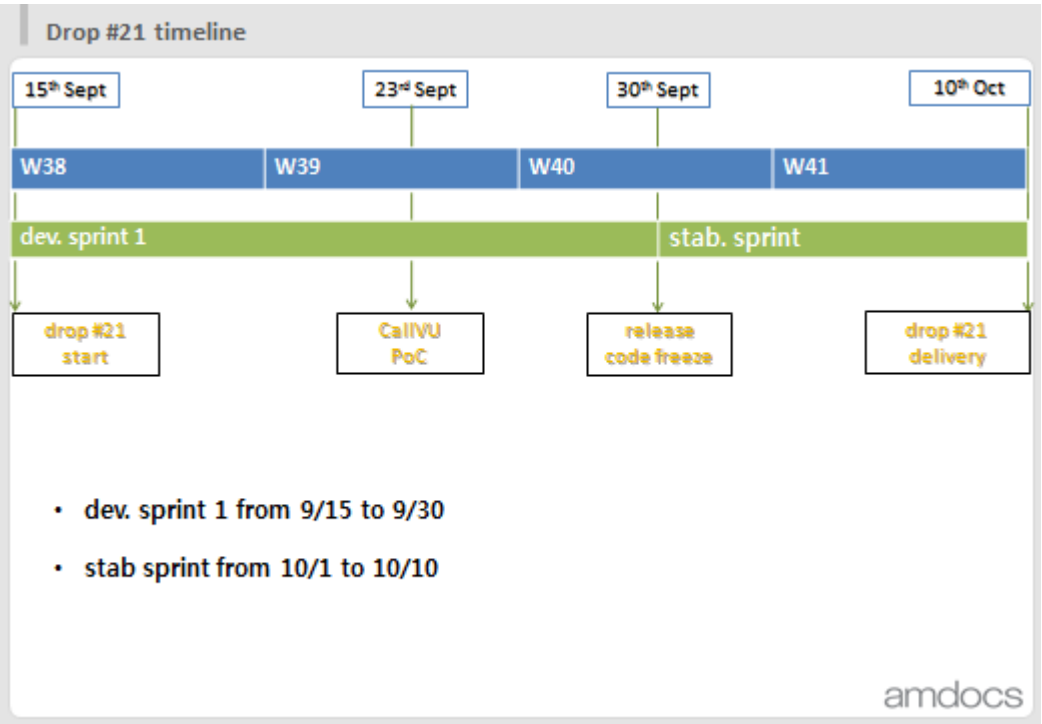


Figure 24 : Drop 21

Le drop 21 a apporté le support d'iOS 8, nouvelle version à date du système d'exploitation mobile d'Apple. De par l'attente forte du support de cette nouvelle version du système d'exploitation mobile d'Apple, le drop ne comporte qu'un sprint de développement pour permettre une livraison rapide d'une nouvelle itération de la suite UXFME. Cela a été l'occasion de faire une stabilisation plus poussée sur l'émulateur UXFME en consacrant l'unique sprint de développement à de la correction de bogues.

3.3.3 Méthodologies agiles et conception

Dans les processus de développement séquentiel comme le cycle en cascade, ou ses dérivés comme le cycle en V, chaque phase a lieu l'une après l'autre. Dans le domaine de l'ingénierie logicielle ce sont généralement des activités de spécification, de conception (architecture), de code puis de test. Dans un cycle séquentiel, à la fin des activités de spécification et de conception sont produits de nombreux documents. Ce n'est qu'à la fin du cycle complet que l'on a un logiciel fonctionnel et testé.

Cette approche est héritée de l'industrie, où elle est nécessaire. Par exemple, quand on produit un véhicule automobile il faut tout d'abord créer des plans précis et sans erreur de conception avant de mettre en place les chaînes de montage et pouvoir produire quoi que ce soit. Dans ce cas, les conséquences d'une modification en amont

du cycle ont un impact majeur (comme la fabrication d'un moule coûteux pour une pièce plastique).

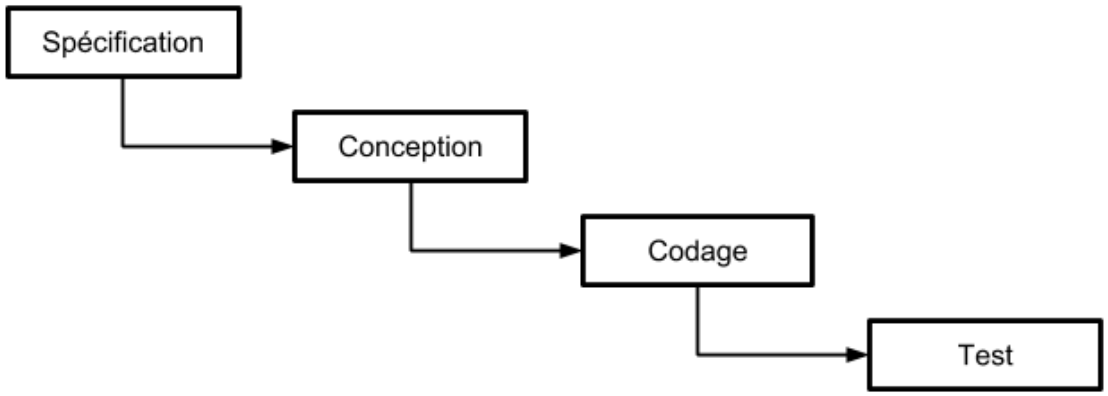


Figure 25 : Modèle en cascade

En théorie, cette approche séquentielle est applicable à tout domaine. Ce modèle idéal reste malheureusement utopique. Dans la réalité, il est courant de revenir sur les activités précédentes. C'est d'ailleurs l'objet du cycle en V qui ajoute au modèle en cascade une vérification à chaque phase, avec retour à l'étape précédente en cas d'anomalie. Scrum et les méthodologies agiles sont plus pragmatiques en rendant les phases de spécification et de conception continues. Cela s'applique facilement au génie logiciel, où il est plus facile d'expérimenter, d'itérer et de remanier le code existant. Cela ne signifie évidemment pas qu'il faut négliger la phase de conception et se lancer dans l'implémentation immédiatement.

Scrum n'évoque pas explicitement de pratiques d'ingénierie, mais il est néanmoins nécessaire de faire de la conception au début du premier sprint d'un nouveau composant. Elle prend souvent la forme d'une architecture logicielle, plutôt que d'une conception détaillée, l'objectif étant de n'oublier aucune étape de travail préliminaire à l'implémentation tout en minimisant la documentation à ce qui est utile uniquement. Le but sera alors d'arriver à une architecture logicielle à la fois modulaire et flexible. La modularité permettra de faciliter le développement de façon itérative et de livrer les fonctionnalités par blocs. La flexibilité est quant à elle nécessaire face aux événements qui pourraient avoir lieu pendant le cycle de vie de l'application : retours du client, changements des spécifications, modifications des technologies, évolutions

futures du produit non encore planifiées… La conception se limite donc à l'architecture, constituant une base qui sera enrichie aux sprints suivants.

3.3.4 Architecture logicielle

L'architecture met en jeu l'émulateur UXFME et ses interactions avec le navigateur Internet, l'application mobile hybride de l'utilisateur et la base de données des appareils.

C'est par le biais du navigateur, et les outils de développement associés, que l'utilisateur pourra interagir avec son application émulée.

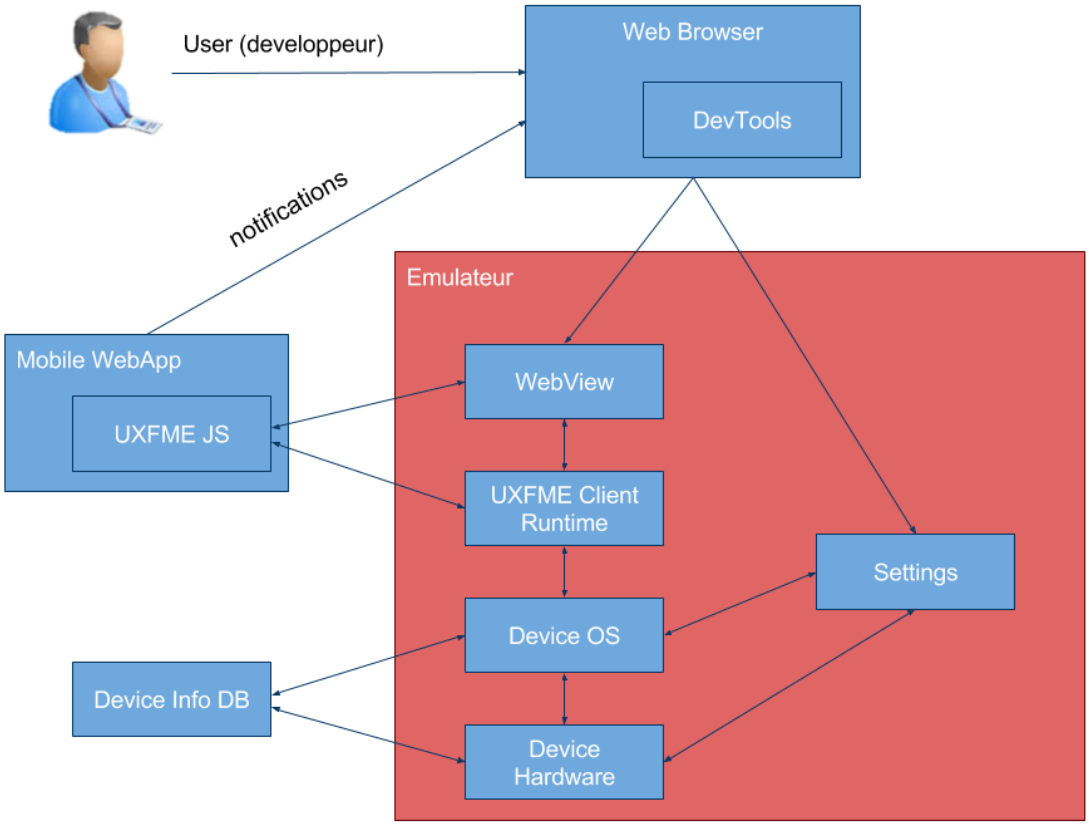


Figure 26 : Architecture haut niveau de l'émulateur UXFME

Le navigateur Internet sera mis à profit pour l'interface utilisateur de l'émulateur et pour ses outils de développement web. Par ailleurs, il n'est pas souhaitable de modifier l'application de l'utilisateur.

L'ensemble des composants internes de l'émulateur et la base de données d'appareils restent à développer.

3.3.5 Composants

Sans entrer dans une conception détaillée peu adaptée à Scrum, il est déjà possible de lister les composants logiques importants répondant aux contraintes de chaque récit utilisateur.

Base de données des appareils UXFME

Pour implémenter la sélection d'appareils mobiles issus de la certification UXFME, il est nécessaire de fournir une base de données. En effet, les rapports de la certification d'appareils UXFME sont à la destination de développeurs d'applications mobiles et sont donc sous forme de documents. Outre le fait que les données présentes dans ces fichiers ne sont pas exploitables directement, elles sont utiles aux développeurs mais pas toutes nécessaires dans le cadre de l'émulateur UXFME.
Cette base de données aura donc pour but de centraliser les caractéristiques utiles à l'émulation. Pour une première itération, cette base de données peut être locale puisqu'uniquement destinée à l'émulateur UXFME, et n'est pas vouée à des mises à jour en temps réel.

Panneau de configuration

Pour configurer les données accessibles depuis l'application émulée ou l'affichage de cette application, il est nécessaire d'offrir un panneau de configuration. Ce panneau doit aussi donner des informations sur l'appareil émulé. Plus généralement et dans un esprit de modularité, chaque fonctionnalité qui en a le besoin doit pouvoir afficher un bloc dans l'interface utilisateur et exposer programmatiquement les données associées.
Par exemple la contrainte de sélection d'appareil mobile parmi ceux présents dans la base de données induit qu'on puisse présenter dans le panneau de configuration un bloc d'interface utilisateur correspondant à cette fonctionnalité.

Mise à jour et notification push

Les fonctions de mise à jour et notification push doivent exposer les réglages associés dans le panneau de configuration.

Sur appareil mobile, la notification push est un événement extérieur à l'application web qui apparaît dans l'interface native du téléphone ou de la tablette. Dans le cadre de l'émulateur une notification ne peut apparaître ni dans la visualisation de l'application ni dans le panneau de configuration.
Récemment les applications pour ordinateur personnel et les navigateurs Internet ont implémenté ce système de notification sur machine de bureau. On pourra alors, si applicable, utiliser le mécanisme de notification du navigateur.

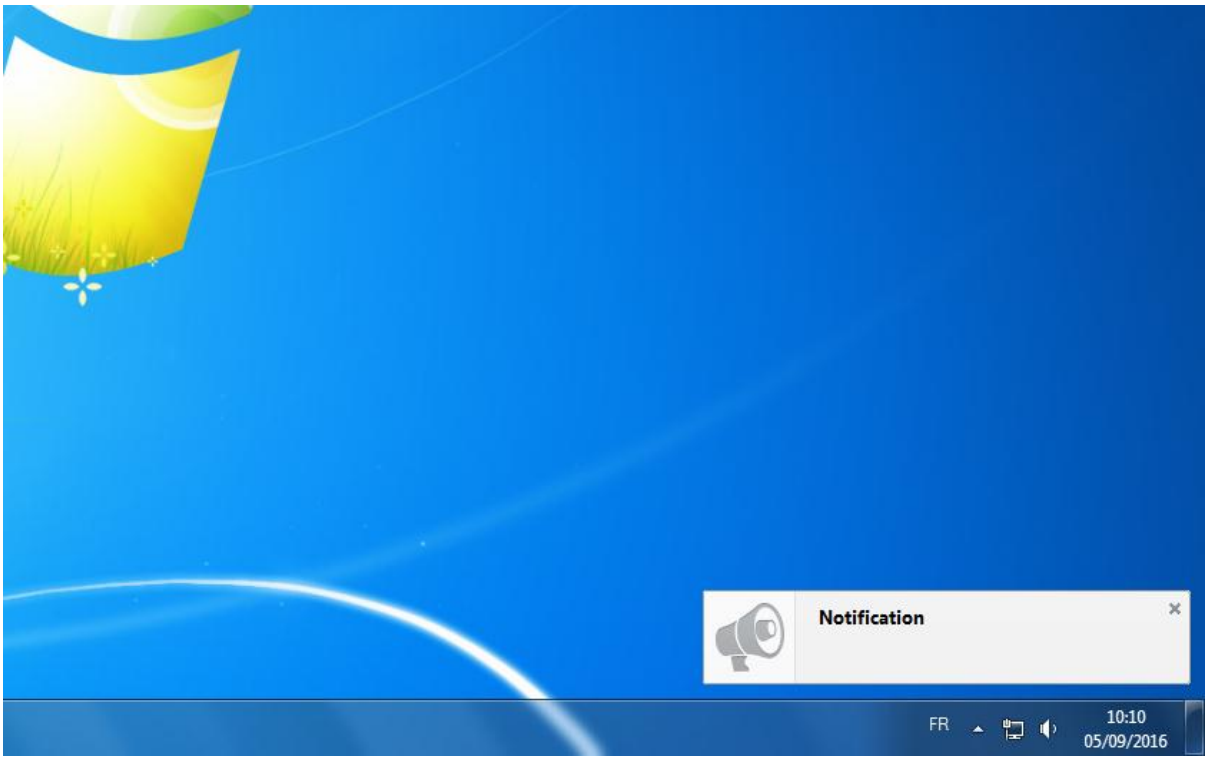


Figure 27 : Notification de Mozilla Firefox sur Windows

Visualisation de l'application

La visualisation de l'application web est effectuée par le navigateur web, se substituant à la WebView pour le rendu graphique. En effet, il faut que l'on puisse utiliser les outils de débogage du navigateur. La visualisation prend en compte la définition et la taille physique de l'écran de l'appareil émulé.

L'objectif du moteur d'exécution de l'émulateur est de reproduire l'exécution des appels au UXFME Client Runtime, rendant ainsi possible l'utilisation des APIs client UXFME.

L'application web, y compris la bibliothèque JavaScript UXFME, ne doit pas être modifiée. Ce sont les seules parties existantes, le reste devra être réalisé. Le comportement du Client Runtime et ses interactions avec la WebView sont partie intégrante de l'émulateur à développer, de même que l'appareil mobile et son système d'exploitation dont les caractéristiques sont obtenues à partir de la base de données des appareils.

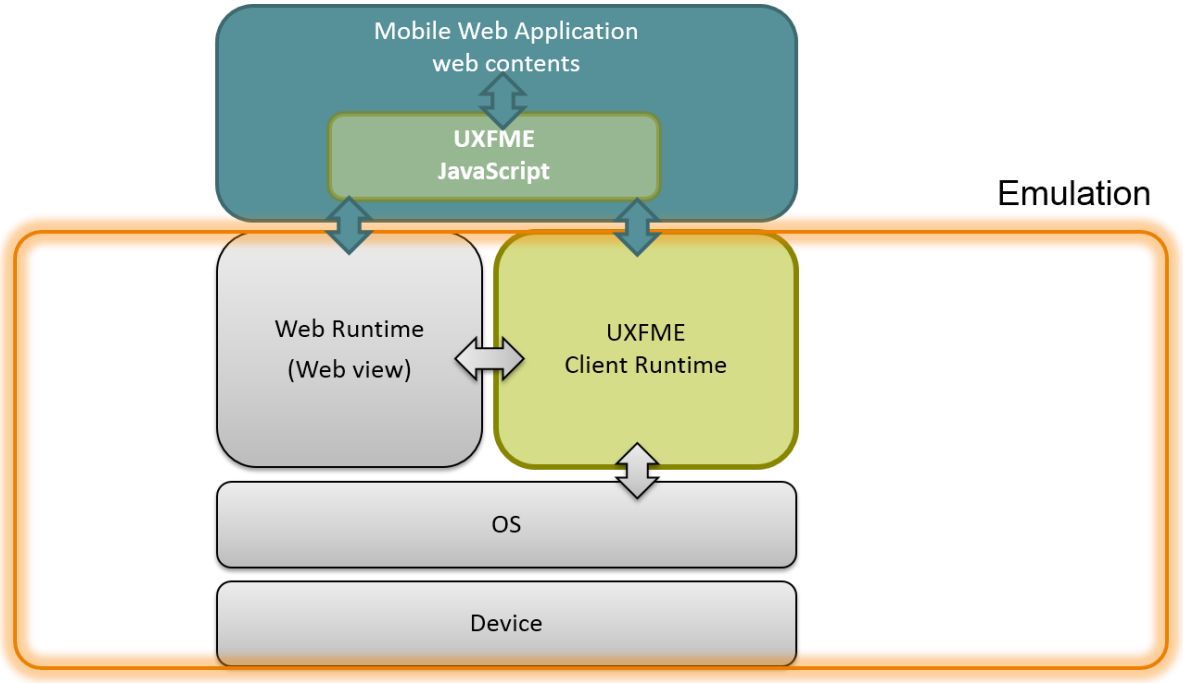


Figure 28 : Client UXFME et prise en charge de l'émulation

A noter que cette représentation du client UXFME et du moteur d'exécution de l'émulateur n'utilise pas de standard particulier. Cette figure n'est présente qu'à titre d'illustration du périmètre devant être couvert par l'émulateur et ne reflète pas l'architecture réelle du client UXFME, qui diffère selon la plateforme mobile.

3.3.6 Ergonomie

L'émulateur UXFME s'inspire de l'expérience utilisateur offerte par Apache Ripple, qui est par ailleurs une approche courante pour un émulateur. Chaque bloc représente des réglages et des informations liés à une fonctionnalité particulière. Par exemple pour la notification push on peut voir un bloc permettant d'éditer les deux données représentant une notification que sont le titre et les données. Ce même bloc permet ensuite l'envoi de la notification.
L'aperçu de l'application émulée est lui visible au centre de l'écran.

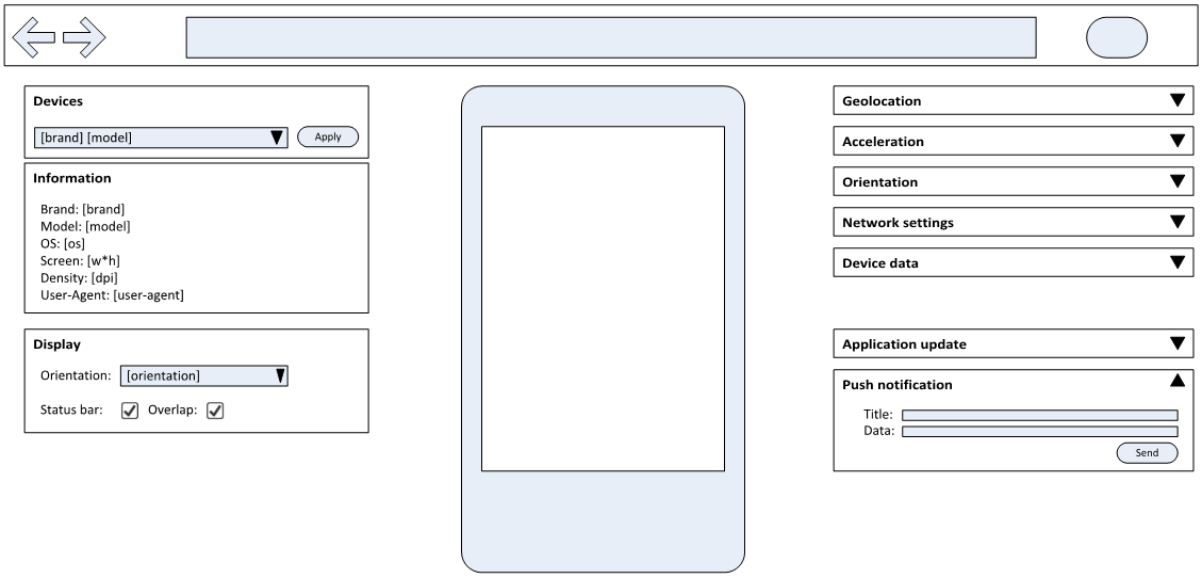


Figure 29 : Schématisation en fil de fer de l'émulateur UXFME

La conception de l'interface utilisateur prend la forme d'une schématisation en fil de fer dite "wireframe". Ce type de schéma représente la mise en page d'un logiciel ou d'une page web sans pour autant rentrer dans le détail du design graphique.
Au sens de Scrum il s'agit d'un premier livrable, sous la forme d'un prototype. En effet, ce schéma permet de valider rapidement l'ensemble de l'expérience utilisateur avec les parties prenantes. Non seulement l'agencement des différentes parties est représenté mais il est aussi possible avec plusieurs schémas de représenter la logique de l'application via un parcours utilisateur.

L’étude des solutions préexistantes a démontré la nécessité de concevoir et de développer l’émulateur UXFME. La conception ayant permis de découper le développement en stories, il est possible de passer à la réalisation.

4 Réalisation

Cette partie décrit la réalisation de la solution découlant de la phase d'étude précédemment décrite.
Tout d'abord seront présentés les choix techniques effectués en fonction des besoins utilisateurs, des résultats de l'étude des solutions préexistantes et de l'environnement de l'entreprise.
Une seconde partie détaillera plus particulièrement les extensions pour Google Chrome, leur mode de distribution et leurs mécanismes de sécurité.
La troisième partie détaille trois aspects importants de l'émulation d'applications mobiles hybrides : la base de données d'appareils, l'émulation de l'affichage et le modèle d'exécution.
Enfin, la réalisation de l'interface utilisateur sera présentée dans une dernière partie.

4.1 Choix techniques

L'environnement technique décrit ci-après est issu de choix raisonnés pris en tenant compte des besoins utilisateurs et des contraintes techniques du projet. Google Chrome s'est par exemple naturellement imposé. Par ailleurs, les briques logicielles libres et open-source sont éprouvées et leurs choix justifiés ci-après, tout comme les outils de développements utilisés.

4.1.1 Google Chrome

Comme évoqué dans le chapitre sur les contraintes techniques, les navigateurs Internet généralement supportés par les produits Amdocs sont Google Chrome et Microsoft Internet Explorer. Le choix s'est naturellement orienté vers Chrome pour plusieurs raisons.
Tout d'abord Chrome offre un meilleur support des standards HTML et CSS, sans doute dû au fait que ce navigateur est plus récent dans sa conception. Il est intéressant de noter qu'il partage le même socle logiciel (WebKit) que la plupart des WebViews des appareils mobiles. Par ailleurs, le moteur d'exécution JavaScript "V8" de Google intégré au navigateur est parmi les plus performants du marché. De plus, il existe plusieurs façons d'étendre le navigateur Chrome, alors qu'Internet Explorer ne propose que ActiveX, une technologie vieillissante souvent critiquée pour ses

problèmes de sécurité. Enfin, Internet Explorer est abandonné par Microsoft au profit du navigateur Edge, que ce soit sur mobile ou ordinateur de bureau. Très prometteur, ce nouveau navigateur n'est disponible que sur Windows 10 et n'est donc pas encore disponible au sein d'Amdocs.

4.1.2 Extensions Chrome

Pour permettre de créer une application locale au poste des développeurs utilisateurs de l'émulateur UXFME, il a été décidé de mettre à profit les possibilités d'extensions de Google Chrome. En effet, il apparaît inutile d'installer un serveur pour une application locale et une simple page HTML est quant à elle trop limitée dans ses possibilités, entre autres par la sandbox de Chrome. En effet le bac à sable de Chrome, un mécanisme qui vise à isoler au maximum le contexte de la page Internet du système d'exploitation, ne permet pas l'accès à un certain nombre de fonctionnalités du navigateur.

Il existe trois façons d'étendre le navigateur Google Chrome : le plugin, l'application et l'extension.
Les plugins Chrome sont spécifiques et ne sont pas adaptés à la création d'une application. Ce sont des plugins avec du code natif généralement destinés à proposer le support d'un format spécifique de contenu. Exemples de plugins : Adobe Flash, le gestionnaire de droits numériques (DRM) Widevine.
Les applications sont des applications web que l'on peut installer au sein du navigateur. Elles peuvent alors proposer du contenu hors-ligne. Ces applications sont lancées via leurs icônes respectives depuis un tiroir d'applications présent dans Chrome, à la manière de ce que l'on retrouve couramment sur un smartphone. Exemples d'applications : Google Drive, Line, Outlook.
Les extensions sont quant à elles destinées à étendre le contenu d'une page web ou les fonctionnalités du navigateur. Elles peuvent se superposer et enrichir les pages visitées et les applications, contrairement aux applications qui elles sont autonomes. Un exemple populaire, bien que controversé sont les extensions de blocage de publicité. Exemples d'extensions : Adblock, IE Tab.

Il a été choisi de développer une extension plutôt qu'une application pour sa capacité de superposition fonctionnelle. En effet, dans son concept l'émulateur vient enrichir la page visitée ou l'application web pour vérifier son bon fonctionnement dans un environnement d'application mobile hybride. C'est par ailleurs le même choix qui a été fait par Apache Ripple. Le choix entre application ou extension n'est au final que peu impactant sur le développement de l'émulateur, les interfaces de programmations disponibles étant extrêmement proches.

4.1.3 FOSS Free and Open-Source Software

Quelques briques logicielles libres et open-source (FOSS) ont été introduites de façon raisonnée durant le développement de l'émulateur UXFME afin de faciliter le développement et de réduire les délais.

Les FOSS suivants ont été utilisés :

- jQuery
- Bootstrap, Glyphicon Halflings
- EJS

Ces composants sont éprouvés et leurs choix justifiés ci-après, tout comme les outils de développements utilisés.

jQuery

jQuery est une bibliothèque JavaScript facilitant entre autres les opérations de recherche et de manipulation d'éléments HTML (via sa représentation : le DOM), la création d'animations ou encore l'utilisation de requêtes serveur asynchrones. La bibliothèque jQuery est un standard de fait dans l'industrie logicielle web, puisqu'elle est présente au coeur d'une majorité de sites Internet. En effet, une étude de 2016 portant sur les sites web les plus populaires montre que jQuery est utilisé dans 70.6% des sites[9].

[9] Usage Statistics and Market Share of JavaScript Libraries for Websites, July 2016 https://w3techs.com/technologies/overview/javascript_library/all

La bibliothèque jQuery a de nombreux avantages, tels que le support de nombreux navigateurs et permet même de diminuer les incompatibilités entre navigateurs. En effet, les différents navigateurs ne supportent pas tous les mêmes fonctionnalités ou à des degrés différents et un même code JavaScript peut ne pas fonctionner sur certains environnements. jQuery est une librairie relativement légère, mais il est également possible de l'étendre par un système de plugins.

La bibliothèque est libre et open-source et distribuée sous une licence MIT, une licence de type permissive. Le développement est open-source mais est supporté par une fondation dont le but est de coordonner le développement et assurer la promotion de jQuery mais aussi d'autres outils populaires comme par exemple Grunt, Lodash, QUnit, RequireJS. Parmi les entreprises membres de la fondation jQuery on trouve de grands acteurs de l'informatique comme IBM, Mozilla, Adobe, Intel[10]…

Ce composant extrêmement populaire facilite souvent le développement d'applications web, et est également un prérequis nécessaire au fonctionnement de Bootstrap. C'est pour ces deux raisons qu'il a été décidé d'introduire jQuery parmi les bibliothèques tierces utilisées dans l'émulateur UXFME.

Bootstrap

Bootstrap est un framework open source créé par Twitter permettant de faciliter le développement de sites et d'applications web en fournissant une collection de formulaires, boutons, icônes, et autres éléments couramment nécessaires à la construction d'une interface utilisateur. Bootstrap est un composant open-source très populaire.

Ce framework offre une très grande collection de composants, compatibles avec les navigateurs Internet majeurs et est pensé d'abord pour le mobile (mobile-first). Par ailleurs, Bootstrap permet d'améliorer l'aspect graphique d'une page HTML par l'utilisation de feuilles de style CSS.

[10] https://jquery.org

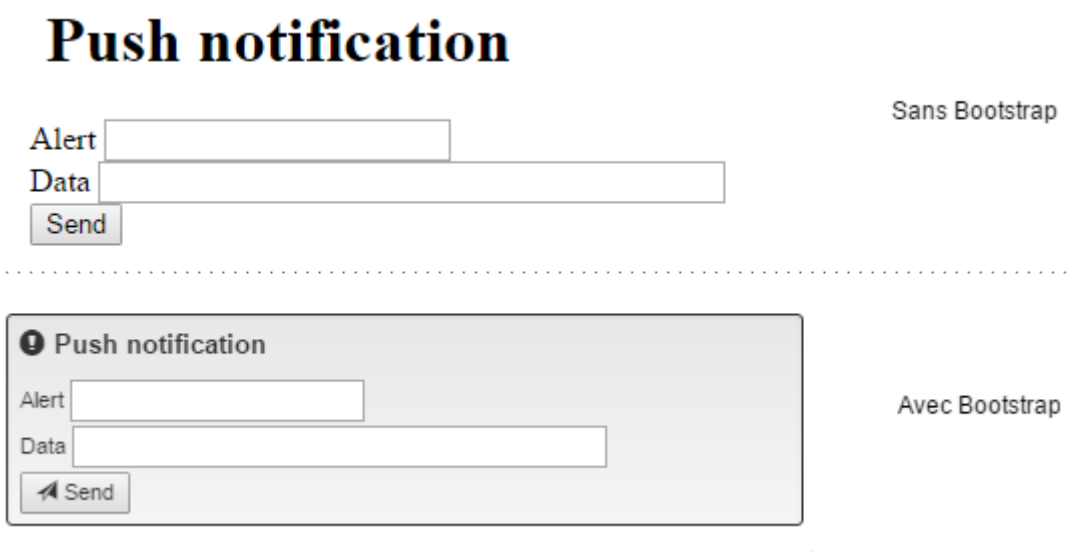


Figure 30 : Formulaire simple stylé avec Bootstrap

Bootstrap propose une police de caractère dont les glyphes, c'est-à-dire les caractères, représentent des icônes : Glyphicons. Ces icônes et symboles sont monochromes et très clairs. L'utilisation d'une police de caractères, vectorielle, permet de mettre à l'échelle ces symboles graphiques sans aucune perte de qualité.

Figure 31 : Glyphicons

Bootstrap est distribué sous une licence de type permissive : la licence MIT. Le développement, open-source, est coordonnée par les créateurs originels du composant qui sont des employés de Twitter.

L'utilisation de Bootstrap dans l'émulateur UXFME permet d'améliorer l'aspect graphique et l'ergonomie rapidement, sans avoir à recourir à des compétences extérieures à l'équipe.

EJS

Embedded JavaScript (EJS) est un système de template côté client. C'est-à-dire qu'il combine des données JavaScript sous forme d'objets ainsi qu'un patron pour produire un contenu HTML. Ce système permet de faire une séparation claire entre données et présentation.

Figure 32 : Exemple d'EJS

Un moteur de template permet d'éviter la génération d'éléments HTML avec JavaScript, ce qui peut être compliqué et surtout peu lisible. En effet, dans certains cas la structure de la page HTML est mélangée avec de la logique écrite en JavaScript voire du traitement de données. Il est aussi plus aisé pour deux personnes de travailler simultanément sur respectivement le design et le développement, EJS permettant un tel découplage. Enfin, EJS supporte la mise en cache du HTML résultant du moteur de template pour de meilleures performances.

Embedded JavaScript est principalement développé par l'entreprise Bitovi qui distribue les sources sous les conditions de la licence MIT. Cette licence est donc compatible avec la politique FOSS d'Amdocs, puisque non virale et permissive.

Ce composant a été introduit pour fournir une couche d'abstraction supplémentaire entre la présentation et les données de l'émulateur, facilitant ainsi le développement. EJS a été choisi pour sa simplicité, mais il est évident que d'autres offres intéressantes existent parmi lesquelles Mustache ou bien HandlebarsJS.

4.1.4 Outils

Les outils utilisés pour le développement de l'émulateur UXFME sont ceux rendus disponibles ou parfois imposés par Amdocs. L'environnement de développement intégré (IDE) est IntelliJ IDEA Ultimate, outil payant fourni par Amdocs. Cet IDE permet de profiter d'une bonne coloration syntaxique et d'une auto-completion avancée pour le JavaScript. D'autres outils comme l'IDE Eclipse ou l'éditeur Sublime Text semblent en deçà sur ces points.

Le système de gestion de configuration est Perforce, utilisé dans l'ensemble d'Amdocs. Ce produit commercial se base sur une architecture centralisée avec un unique dépôt.

L'émulateur UXFME utilise également le linter JSHint. Le terme "linter", néologisme anglais, désigne un outil s'inspirant du logiciel Unix lint. C'est un utilitaire procédant à de l'analyse statique de code source pour y déceler des bogues éventuels, ou de mauvaises pratiques de programmation. JSHint permet donc d'améliorer effectivement la qualité du code en donnant des indices sur de possibles bogues ou des mauvaises pratiques. Par exemple, en JavaScript : utilisation de l'opérateur d'égalité faible, variable non initialisée, variable non utilisée, globales, utilisation de l'instruction eval…

4.2 Extensions Chrome

Les extensions de Google Chrome permettent d'ajouter ou de modifier des fonctionnalités du navigateur Internet et de modifier les pages qui sont visitées. Le développement se fait grâce aux technologies web classiques c'est-à-dire HTML, CSS et JavaScript. Des APIs spécifiques permettent par ailleurs d'interagir avec les pages web ou d'accéder à certaines fonctionnalités du navigateur.

Les extensions Chrome sont de plus compatibles avec les navigateurs Opera et Microsoft Edge et en grande partie compatibles avec les WebExtensions de Mozilla Firefox[11].

4.2.1 Aperçu

Une extension est généralement constituée d'une page d'arrière-plan unique, qui s'exécute dans un contexte séparé, et de scripts de contenu qui eux peuvent interagir avec la page web affichée dans le navigateur Internet.

Les scripts de contenu et la page d'arrière-plan sont dans des contextes d'exécution différents, néanmoins ces scripts peuvent communiquer à l'aide d'un système de messages. Ce mécanisme est d'ailleurs largement utilisé dans l'émulateur UXFME.

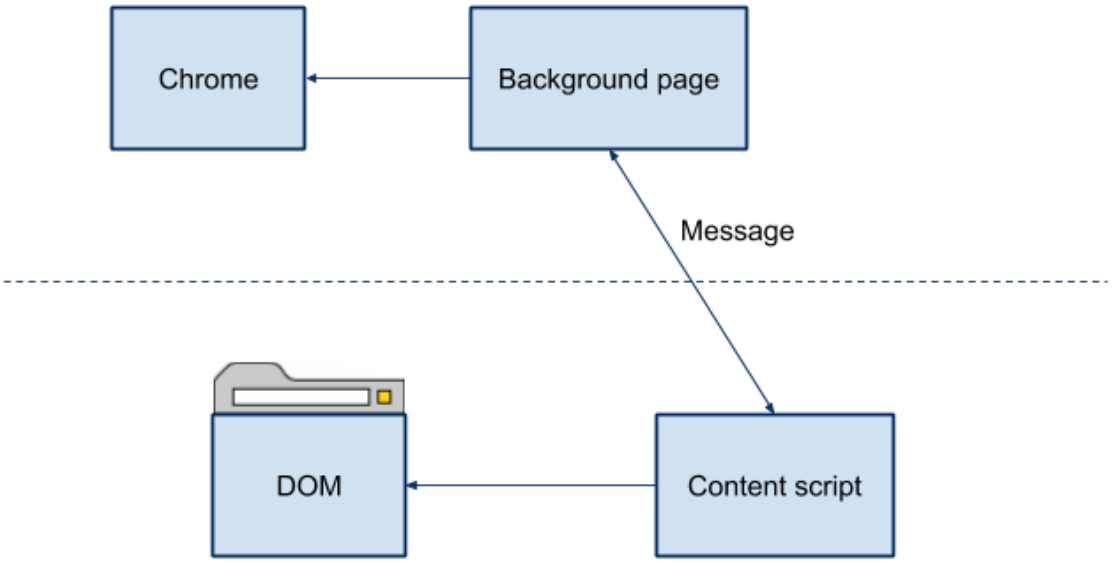


Figure 33 : Vue d'ensemble d'une extension Chrome

[11] https://developer.mozilla.org/en-US/Add-ons/WebExtensions

Le fichier manifeste

Le manifeste est un fichier central décrivant l'extension Chrome. C'est ce fichier de méta-données qui énumère les propriétés caractérisant l'extension comme par exemple son nom ou sa version. On y trouve également des propriétés liées aux droits d'accès et aux permissions. Il y est aussi décrit l'ensemble des dossiers ou fichiers constituant l'extension.
L'ensemble de ces informations permettent à Chrome d'instancier l'extension.

Voici le fichier manifeste de l'émulateur UXFME :

```
{
  "manifest_version": 2,
  "name": "UXFME Device Emulator 10.0.1",
  "version": "28.0.2",
  "description": "UXFME Device Emulator Chrome Extension",
  "icons": {
    "16": "images/icon16.png",
    "48": "images/icon48.png",
    "128": "images/icon128.png"
  },
  "browser_action": {
    "default_icon": "images/icon48.png"
  },
  "background": {
    "scripts": ["background.js", "emulation/deviceapis-emulation.js"
    ]
  },
  "content_scripts": [
    {
      "matches": ["http://*/*", "https://*/*", "file://*/*"],
      "run_at": "document_start",
      "js": ["libs/ejs_production.js", "contentscript.js"],
      "all_frames": true
    }
  ],
  "permissions": [
    "geolocation",
    "notifications",
    "tabs",
    "webNavigation",
    "webRequest",
    "webRequestBlocking",
    "<all_urls>"
  ],
  "web_accessible_resources": [
    "views/ecp.ejs",
    "views/help.pdf",
    "views/js/jquery-1.11.1.min.js",
    "views/js/bootstrap.min.js",
    "views/css/bootstrap.min.css",
    "views/css/bootstrap-theme.min.css",
    "views/css/ecp.css",
    "emulation/dialog.css",
    "views/fonts/glyphicons-halflings-regular.eot",
    "views/fonts/glyphicons-halflings-regular.svg",
    "views/fonts/glyphicons-halflings-regular.tff",
    "views/fonts/glyphicons-halflings-regular.woff",
    "devicelist.json"
  ]
}
```

Figure 34 : Manifeste de l'extension Chrome émulateur UXFME

On y retrouve :

- La version du format de manifeste
- Le nom de l'extension, qui sera présenté à l'utilisateur
- Sa version
- Une description, qui sera présentée à l'utilisateur

- Une série d’icônes qui seront utilisées dans le Chrome Web Store et dans le navigateur
- Browser action, détaillé ci-après
- Les pages de background, détaillé ci-après
- Les pages de content script, détaillé ci-après
- Les permissions et la liste blanche de ressources accessibles sont détaillés dans le chapitre sécurité

L’arborescence des fichiers constituants l’extension émulateur UXFME reflète ce qui est déclaré dans le fichier manifeste. Par ailleurs, on notera la présence des FOSS EJS, Bootstrap et jQuery.

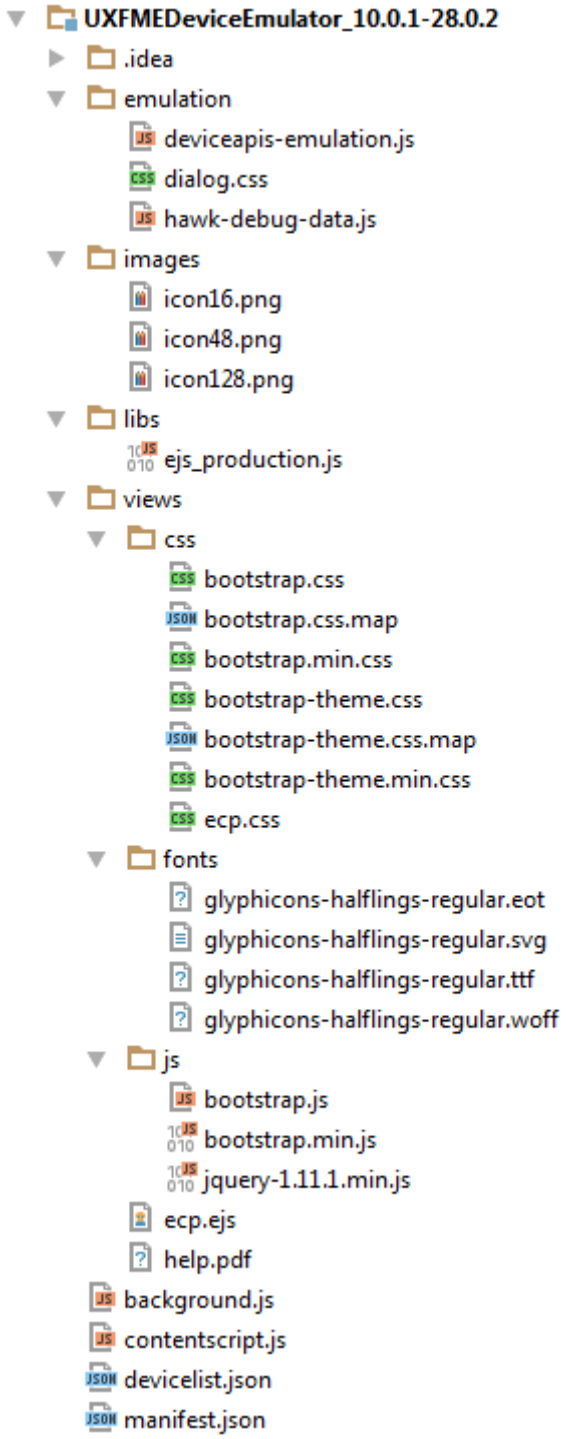


Figure 35 : Arborescence des fichiers constituant l'émulateur UXFME

Browser action

Une action de navigateur est représentée sous la forme d'un bouton qui peut être ajouté à la barre d'outils du navigateur. Une extension ne peut avoir qu'un unique bouton d'action, qui est utilisé pour interagir avec l'extension. Cela peut être par exemple mis à profit pour activer ou désactiver l'extension, ou ouvrir un menu sous

forme de popup. Ces popups sont définies sous forme de HTML, CSS et JavaScript, comme une page web classique.
Selon le fait qu'une popup est définie ou non, les événements liés au bouton sont soit gérés par la popup, soit par la page d'arrière-plan.

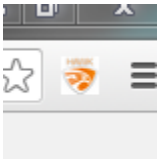


Figure 36 : Le bouton d'action de l'émulateur UXFME

Background

La page d'arrière-plan est une page HTML unique et propre à l'extension, quel que soit le nombre de pages web ouvertes dans le navigateur Internet. Les scripts associés à cette page, appelés scripts d'arrière-plan, sont à tout moment disponibles indépendamment du cycle de vie des pages visitées. Il est d'ailleurs possible de seulement définir un script d'arrière-plan ; la page HTML sera quant à elle créée automatiquement.
Les scripts d'arrière-plan sont chargés dès que l'extension est instanciée et le restent jusqu'à ce que le navigateur soit fermé, l'extension désactivée ou désinstallée. Les scripts d'arrière-plan ont accès à l'ensemble des APIs offertes par Chrome pour la création d'extension.
Ces APIs offrent de nombreuses possibilités. Par exemple : gérer les marque-pages, gérer l'historique, ajouter des menus contextuels, bloquer des requêtes réseau...

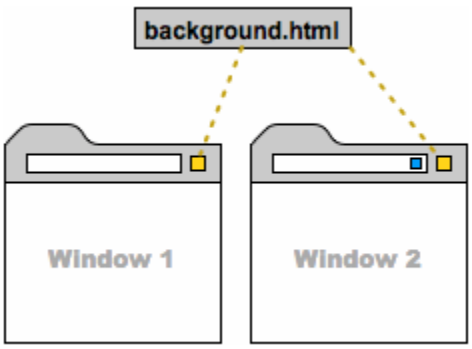


Figure 37 : La page d'arrière-plan est unique

Cette page et les scripts associés sont dans un contexte d'exécution dédié à l'extension, sans accès direct aux pages ouvertes dans le navigateur. Il faut alors pour communiquer avec les différents onglets faire appel aux APIs d'envoi de messages via les scripts de contenus.

Content script

Les scripts de contenus, comme leur nom l'indique, sont des scripts pouvant interagir avec le contenu de la page web affichée dans un onglet. Le script de contenu peut interagir avec le DOM de la page visitée, mais ne s'exécute pas dans le même contexte d'exécution JavaScript.

En effet, pour des considérations de sécurité, les scripts de contenu s'exécutent dans un environnement spécial appelé un monde isolé (isolated world). Ils ont accès au DOM de la page, mais pas aux variables ou fonctions JavaScript de cette page. L'inverse est aussi vrai : la page web visitée n'a quant à elle pas accès en JavaScript aux scripts de contenu. Les scripts des différentes extensions sont également isolés les uns des autres.

Il n'est possible d'utiliser que quelques APIs destinées aux extensions Chrome, contrairement aux scripts d'arrière-plan. Pour contourner cette limitation, il est possible d'utiliser le système de messagerie pour envoyer un message au script d'arrière-plan. Le script d'arrière-plan pourra lui utiliser une API inaccessible au script de contenu.

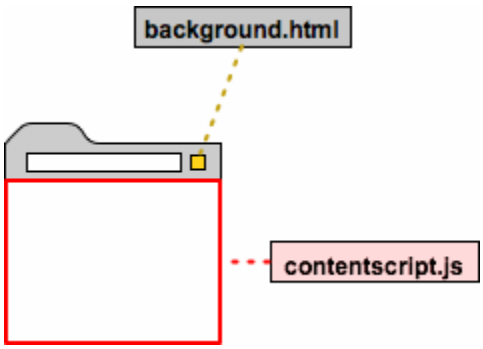


Figure 38 : Un content script peut agir sur le DOM d'une page

4.2.2 Modes de distribution

Les applications et extensions Chrome peuvent être distribuées via un magasin d'application, le Chrome Web Store, ou sous la forme d'un fichier au format CRX.

Magasin d'application Chrome Web Store

Chrome Web Store est le magasin d'applications dédié à Chrome et aux ordinateurs Chromebook. Par son biais sont distribués applications, extensions ou encore thèmes pour le navigateur. Le processus de publication, contrairement aux magasins d'applications pour mobiles, est relativement simple et rapide. Il est par ailleurs possible de restreindre l'accès de l'application ou de l'extension à un groupe de personnes, fonctionnalité utile par exemple durant une phase de test [3].

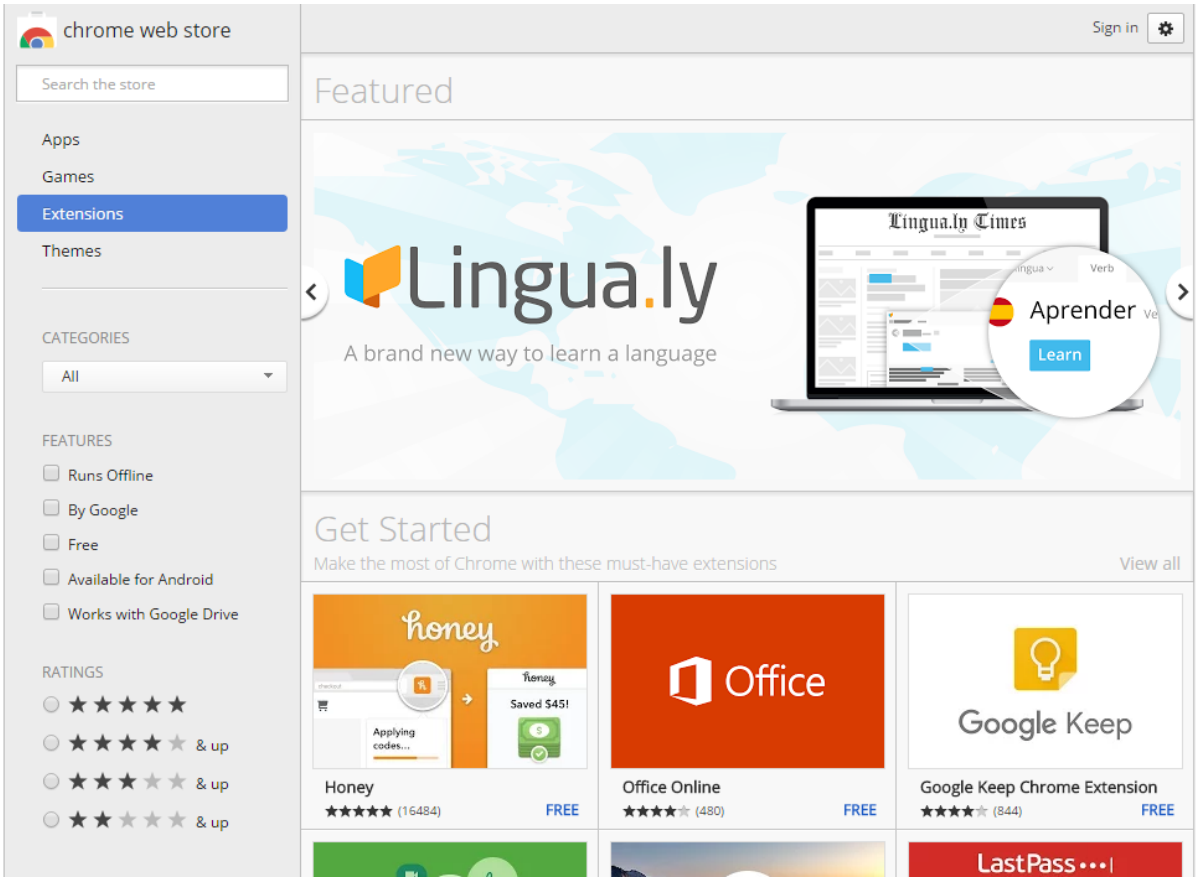


Figure 39 : Chrome Web Store

L'application a un numéro d'identification unique et est liée avec un compte développeur. La provenance d'une application ou d'une extension distribuée est donc

garantie à la fois par le Chrome Web Store et par la signature utilisée par le développeur au moment de la publication.

Packaging d'extension : le format CRX

Pour publier une extension Chrome en dehors du web store, il faut créer un fichier de type CRX qui est un fichier compressé ZIP avec un entête spécial et une extension de fichier .crx.
L'entête contient la clef publique de l'auteur et la signature de l'extension. Plus précisément, la signature est générée depuis le fichier ZIP contenant les fichiers de l'extension. La signature est créée via la fonction de hachage SHA-1 à l'aide de la clef privée de l'auteur de l'extension. Cette signature garantit l'intégrité du contenu de l'extension compressée.

```
43 72 32 34   # "Cr24" -- indicateur de format
02 00 00 00   # 2 -- numéro de version du format CRX
A2 00 00 00   # 162 -- longueur de la clef publique en octets
80 00 00 00   # 128 -- longueur de la signature en octets
..........    # contenu de la clef publique
..........    # contenu de la signature
..........    # données du fichier ZIP
```

Figure 40 : Fichier CRX en hexadécimal

En mode développeur, il est possible depuis Chrome d'exporter manuellement une extension Chrome en format CRX. Néanmoins, cette phase manuelle n'est pas adaptée aux méthodes de travail de l'équipe UXFME. Un logiciel utilitaire a donc été développé afin d'automatiser la création de fichier d'installation d'extension Chrome, utilisable notamment dans le cadre de l'intégration continue.

4.2.3 Sécurité

En plus de la signature et éventuellement de la distribution par le Chrome Web Store qui garantissent la provenance de l'extension, deux autres mécanismes améliorent la sécurité des utilisateurs d'extension : les permissions et la liste blanche.

Permissions

Pour l'utilisation de certaines APIs fournies par Chrome, il est nécessaire de déclarer des permissions dans le fichier manifeste de l'application ou de l'extension. Ce

système permet de limiter ce qui est accessible par une extension, et affiche un message d'avertissement clair à l'utilisateur final au moment de l'installation.

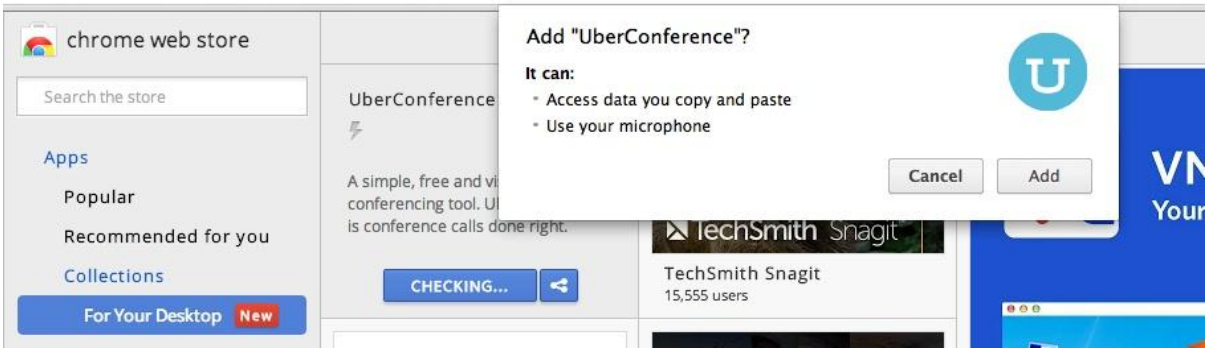


Figure 41 : Boîte de dialogue affichant les permissions demandées

Par exemple, l'utilisateur sera notifié qu'une extension pourra "Utiliser votre microphone" si l'on déclare la permission audio dans le manifeste. Ce mécanisme est similaire aux pratiques courantes sur plateformes mobiles, notamment sur Android et iOS.

L'émulateur UXFME utilise plusieurs API nécessitant la déclaration de permissions comme l'API de géolocalisation, de notification ou de contrôle des requêtes (pour la surcharge du User-Agent).

Liste blanche des ressources

Il est possible de rendre accessibles des ressources web (HTML, JavaScript, images…) depuis le contexte de la page web consultée. Par défaut, les ressources d'une extension ne sont pas accessibles depuis une page web, ce qui n'était pas le cas dans les précédentes versions de Chrome et de ses extensions. Il était alors possible pour un site mal intentionné de détecter précisément les extensions installées et ainsi suivre l'utilisateur sans l'utilisation de cookies. Il était aussi possible d'exploiter les éventuelles vulnérabilités des extensions.

Les ressources accessibles sont donc définies dans une liste blanche. Une liste blanche liste tout ce qui est accessible, et exclut donc tout le reste. Ce mécanisme est à priori plus efficace ou tout du moins plus simple que celui de liste noire. La liste noire, quant à elle, vise à lister les pages interdites, ce qui en effet est difficile à faire de façon exhaustive.

Cette liste blanche de ressources web accessibles est définie dans le manifeste dans la section “web_accessible_ressources”.

```
35    "web_accessible_resources": [
36      "views/ecp.ejs",
37      "views/help.pdf",
38      "views/js/jquery-1.11.1.min.js",
39      "views/js/bootstrap.min.js",
40      "views/css/bootstrap.min.css",
41      "views/css/bootstrap-theme.min.css",
42      "views/css/ecp.css",
43      "emulation/dialog.css",
44      "views/fonts/glyphicons-halflings-regular.eot",
45      "views/fonts/glyphicons-halflings-regular.svg",
46      "views/fonts/glyphicons-halflings-regular.tff",
47      "views/fonts/glyphicons-halflings-regular.woff",
48      "devicelist.json"
49    ]
```

Figure 42 : Liste blanche de ressources

Les ressources utiles au panneau de configuration de l’émulateur UXFME sont déclarées dans cette liste blanche, comme par exemple le fichier d’aide views/help.pdf.

4.3 Émulation

Durant l’implémentation de l’émulateur UXFME, des problématiques d’ordre technique sont apparues. Certaines d’entre elles sont liées à UXFME et à son fonctionnement interne, d’autres sont profondément liées au domaine de l’émulation ou aux spécificités du développement mobile.
Une implémentation factice de toutes les APIs UXFME a été réalisée, permettant de garder intact le modèle d’exécution des applications mobiles hybrides. Ces APIs ont été implémentée à l’aide de HTML5 quand cela était pertinent.
Cette partie explique en détails certains cas intéressants d’émulation d’appareils mobiles, en introduisant quand nécessaire des notions importantes dans le monde de la mobilité.

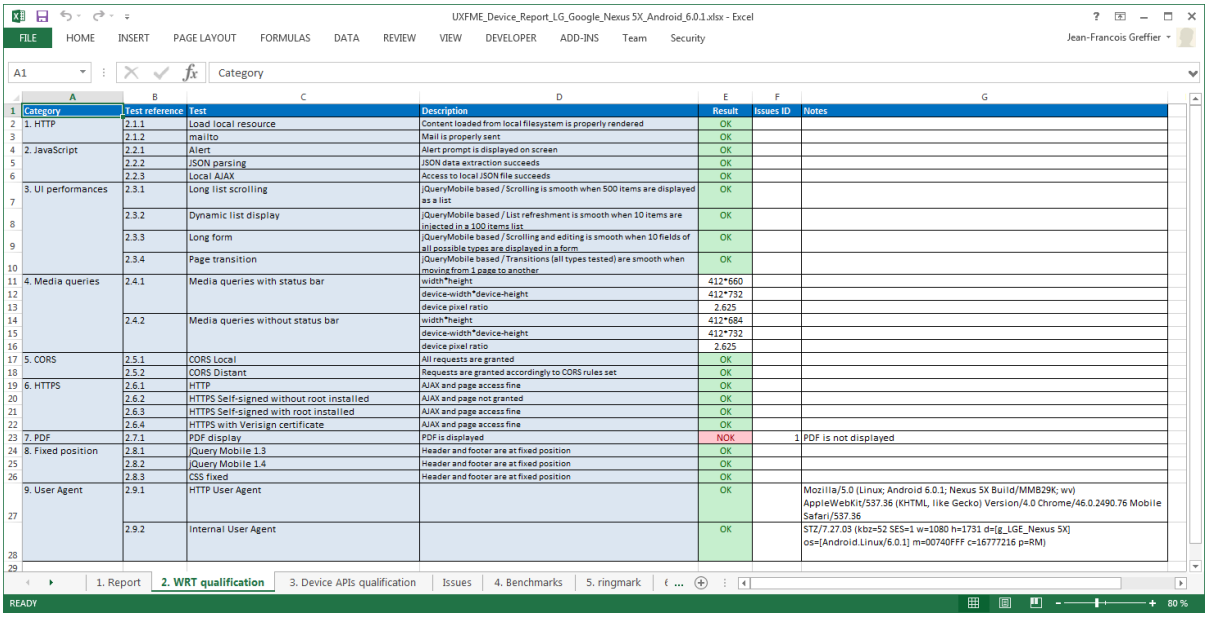
4.3.1 Base de données d’appareils

Comme évoqué dans la partie étude, il apparaît qu’une base de données des appareils à émuler et de leurs caractéristiques est nécessaire. Les rapports de certifications d’appareils UXFME sont sous forme de document et comprennent de

nombreuses données issues des tests de certifications, néanmoins seules certaines de ces données sont utiles dans le cadre de l'émulateur UXFME.

Les rapports de certifications UXFME

Le rapport de certification UXFME d'un appareil mobile est le résultat de caractéristiques constructeurs collectées et agrégées, de résultats de test internes et de tests tierces.



UXFME_Device_Report_LG_Google_Nexus 5X_Android_6.0.1.xlsx - Excel

Category	Test reference	Test	Description	Result	Issues ID	Notes
1. HTTP	2.1.1	Load local resource	Content loaded from local filesystem is properly rendered	OK		
	2.1.2	mailto	Mail is properly sent	OK		
2. JavaScript	2.2.1	Alert	Alert prompt is displayed on screen	OK		
	2.2.2	JSON parsing	JSON data extraction succeeds	OK		
	2.2.3	Local AJAX	Access to local JSON file succeeds	OK		
3. UI performances	2.3.1	Long list scrolling	jQueryMobile based / Scrolling is smooth when 500 items are displayed as a list	OK		
	2.3.2	Dynamic list display	jQueryMobile based / List refreshment is smooth when 10 items are injected in a 100 items list	OK		
	2.3.3	Long form	jQueryMobile based / Scrolling and editing is smooth when 10 fields of all possible types are displayed in a form	OK		
	2.3.4	Page transition	jQueryMobile based / Transitions (all types tested) are smooth when moving from 1 page to another	OK		
4. Media queries	2.4.1	Media queries with status bar	width*height	412*660		
			device-width*device-height	412*732		
			device pixel ratio	2.625		
	2.4.2	Media queries without status bar	width*height	412*684		
			device-width*device-height	412*732		
			device pixel ratio	2.625		
5. CORS	2.5.1	CORS Local	All requests are granted	OK		
	2.5.2	CORS Distant	Requests are granted accordingly to CORS rules set	OK		
6. HTTPS	2.6.1	HTTP	AJAX and page access fine	OK		
	2.6.2	HTTPS Self-signed without root installed	AJAX and page not granted	OK		
	2.6.3	HTTPS Self-signed with root installed	AJAX and page access fine	OK		
	2.6.4	HTTPS with Verisign certificate	AJAX and page access fine	OK		
7. PDF	2.7.1	PDF display	PDF is displayed	NOK	1	PDF is not displayed
8. Fixed position	2.8.1	jQuery Mobile 1.3	Header and footer are at fixed position	OK		
	2.8.2	jQuery Mobile 1.4	Header and footer are at fixed position	OK		
	2.8.3	CSS fixed	Header and footer are at fixed position	OK		
9. User Agent	2.9.1	HTTP User Agent		OK		Mozilla/5.0 (Linux; Android 6.0.1; Nexus 5X Build/MMB29K; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/46.0.2490.76 Mobile Safari/537.36
	2.9.2	Internal User Agent		OK		STZ/7.27.03 (kbz=52 SES=1 w=1080 h=1731 d=[g_LGE_Nexus 5X] os=[Android.Linux/6.0.1] m=00740FFF c=16777216 p=RM)

Figure 43 : Rapport de certification UXFME

Une première partie comprend des détails sur le rapport lui-même (date, testeur, version de la suite de test utilisée) mais aussi des détails d'identification de l'appareil : marque, modèle, nom et version du système d'exploitation.

La deuxième partie est dédiée au niveau de support des API HTML5 et aux informations provenant de la WebView, le composant qui sert de support pour l'exécution des applications mobiles hybrides. Sont par exemple testés le bon fonctionnement d'APIs liées au format JSON (JavaScript Object Notation), au requêtes HTTP asynchrones, aux contrôles d'accès HTTP ou HTTPS… Sont aussi collectées la chaîne de User-Agent, la définition de l'écran avec et sans barre de statut, et d'autres indications sur l'affichage.

La troisième partie indique le niveau de support pour l'ensemble des APIs clientes UXFME : pour chaque fonctionnalité manquante est indiqué s'il s'agit d'une limitation

de l'appareil, d'une limitation de l'implémentation actuelle ou un défaut. S'il s'agit d'un défaut, une référence au système de suivi de bogues de UXFME est présente. Cette vue permet d'avoir une information complète sur quelles APIs UXFME sont utilisables ou non par une application mobile hybride sur l'appareil mobile certifié.

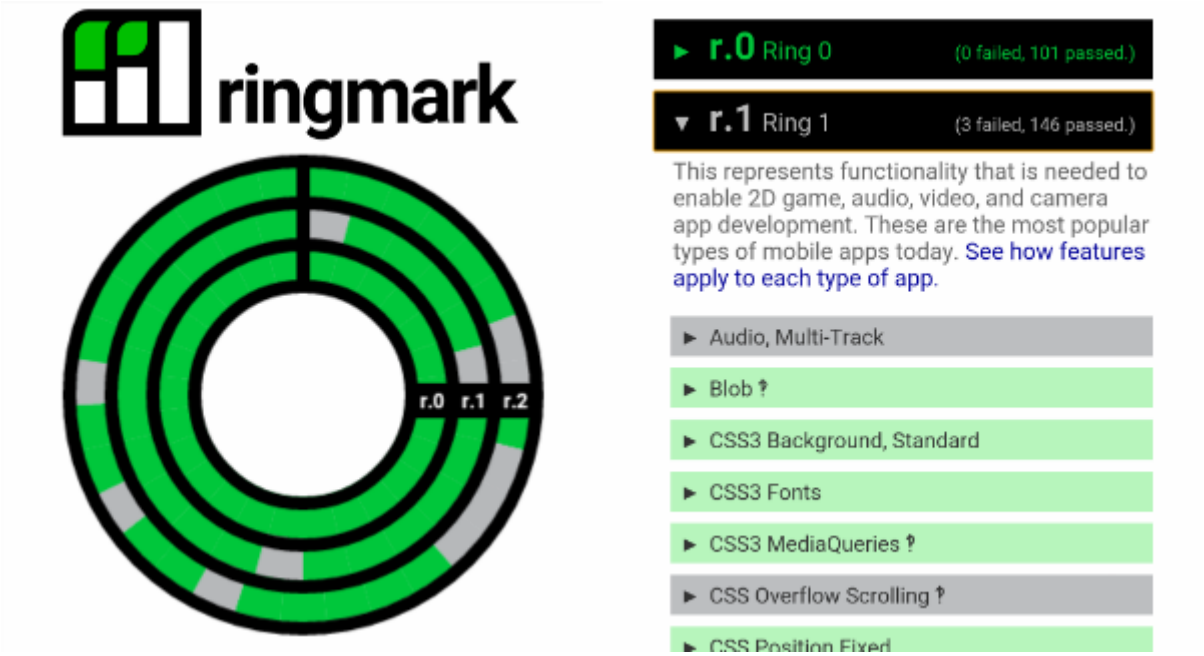


Figure 44 : Ringmark

Enfin, plusieurs informations sur le support des standards HTML5 tels que définis par le W3C sont disponibles. Le World Wide Web Consortium est l'organisme en charge de la standardisation de HTML et CSS mais aussi entre autres des formats XML, PNG, SVG ou même SOAP. Pour ce faire, deux bancs de tests tierces sont utilisés : ringmark et HTML5test. Ringmark est une suite de tests automatisés très populaire créée par Facebook pour tester principalement les fonctions liées à l'Internet mobile. HTML5test est une suite de tests généralistes testant non seulement des fonctionnalités HTML5 comme définies par le W3C, mais aussi de nouvelles fonctions qui ne sont pas encore standardisées ou standardisées par d'autres organismes (comme l'API 3D WebGL, définie par Kronos).

Les données utiles

La structure de donnée représentant un appareil est limitée à ce qui est seulement utile et pertinent dans le cadre de l'émulation. Le support précis, par appareil, des APIs clientes UXFME et des APIs HTML5 sortent donc pour le moment de ce cadre.

En effet, l'émulateur n'as pas comme objectif de simuler les appareils dans leur entièreté.
Par exemple, pendant leurs certifications respectives, HTML5test a révélé que les Motorola Moto G 3ème génération et Samsung Galaxy Note 3 n'avaient pas le même niveau de support de l'API HTML5 de synthèse vocale et de reconnaissance "Web Speech API". L'appareil Motorola ne supporte pas cette API expérimentale, alors que l'appareil Samsung, si.
Il est d'ailleur à noter que le Moto G supporte probablement l'implémentation Webkit de l'API. C'est donc au développeur d'application utilisant UXFME de vérifier cette supposition et de tester sur un appareil réel, si toutefois cette fonctionnalité est utilisée dans son application.

Les données utiles sont donc liées aux caractéristiques identifiant l'appareil, aux données de User-Agent, et aux informations sur l'affichage.
Il faut pour identifier un appareil : sa marque, son modèle et son système d'exploitation (avec sa version).
La chaîne de User-Agent est également extraite des rapports de certification.
Enfin la définition de l'écran en pixels ainsi que la taille de la barre de statut, différente selon le constructeur et/ou le modèle, sont nécessaires pour émuler l'affichage efficacement. La notion de pixelratio, donnant une indication sur la résolution de l'écran, est quant à elle détaillée dans le chapitre sur l'émulation de l'affichage.

Implémentation et format

Pour une première itération, et dans l'esprit des méthodologies agile qui favorisent les logiciels fonctionnels au plus vite, il a été décidé que cette base de données serait locale. En effet, elle est dédiée à l'émulateur UXFME et il n'est pas nécessaire de mettre à jour les données en temps réel.
Les données sont au format JSON ce qui permettra si besoin dans une itération future de basculer par exemple vers une BDD de type NoSQL, éventuellement au travers d'un service Web. Le JSON, JavaScript Object Notation, est un format d'échange de données populaire basé sur la syntaxe du JavaScript. Ce format, créé par Douglas Crockford, est standardisé par la RFC 7159[12] de l'Internet Engineering Task Force et

[12] https://tools.ietf.org/html/rfc7159

par Ecma International (ECMA-404). Le JSON représente une alternative intéressante à d'autres formats d'échanges comme le XML, de par sa nature compacte et sa lisibilité. De plus, de nombreuses bibliothèques utilitaires existent sur une grande variété de langages de programmation et de plateformes pour traiter des données en JSON, qui est d'ailleurs indépendant de JavaScript dont il n'utilise que la notation.
Le format choisi pour représenter un appareil mobile dans la base de données est le reflet des données utiles précédemment identifiées.

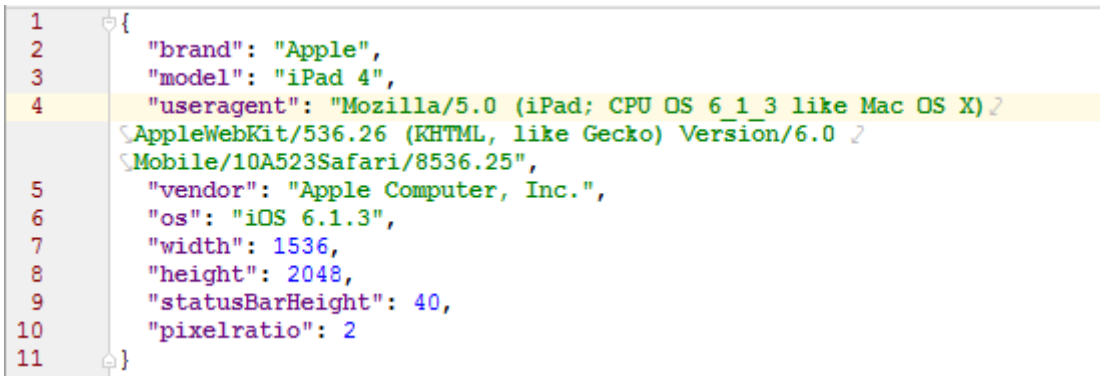


Figure 45 : Enregistrement d'un appareil dans la Base de données

4.3.2 Émuler l'affichage

Gérer les différentes caractéristiques de l'affichage des appareils est crucial pour le design et le développement d'une application mobile. Après une explication des termes définition et résolution, cette partie détaille les spécificités du développement sur mobile et la solution des pixels logiques de CSS. Enfin sera détaillée l'implémentation mise au point pour émuler une application en tenant compte de ces notions comme contraintes de la partie graphique.

Définition et résolution

Les notions de définition et de résolution sont souvent confondues, pourtant elles sont indispensables à la bonne compréhension des problématiques de densité et d'adaptation aux différentes plateformes exécutant une application web, mobile hybride ou native.

La définition désigne la taille en pixels d'un contenu. Les pixels, de l'anglais PICture ELement, désignent les points élémentaires constituant une image.

La résolution est le nombre de points que l'on peut mettre sur un espace physique. Le plus souvent on l'exprime en points par pouce, ou DPI (Dots Per Inches). La taille physique d'un écran et le nombre de pixels qu'il pourra afficher conditionnera donc sa résolution.

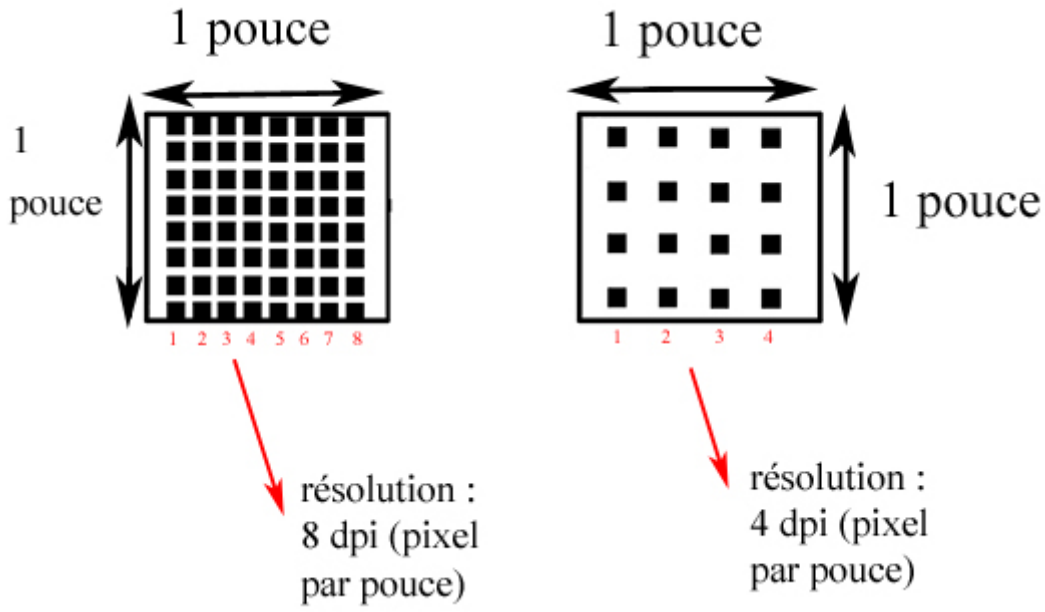


Figure 46 : Illustration de la définition

Par exemple :
L'Apple iPhone 3GS a une définition de 320 pixels de large pour 480 pixels de haut. L'écran fait 3.5 pouces de diagonale avec un ratio d'aspect de 2:3, c'est-à-dire que la largeur de l'écran représente deux tiers de la hauteur. L'écran fait 1.94 pouces de large pour 2.91 pouces de haut.
Le calcul de la résolution de cet appareil est comme suit :

$$\frac{480px}{2.91in} \simeq 165\ DPI$$

Son successeur l'Apple iPhone 4 a une définition de 640*960 pixels, c'est-à-dire le double en largeur et hauteur (pour quatre fois plus de pixels). L'écran quant à lui garde la même taille de 3.5 pouces de diagonale et les mêmes proportions.
La résolution de l'iPhone 4 est donc :

$$\frac{960px}{2.91in} \simeq 330\ DPI$$

Les deux appareils ont des écrans de même taille physique, mais la résolution a doublé. Cette résolution plus élevée est désignée par Apple sous le nom commercial d'écran retina.

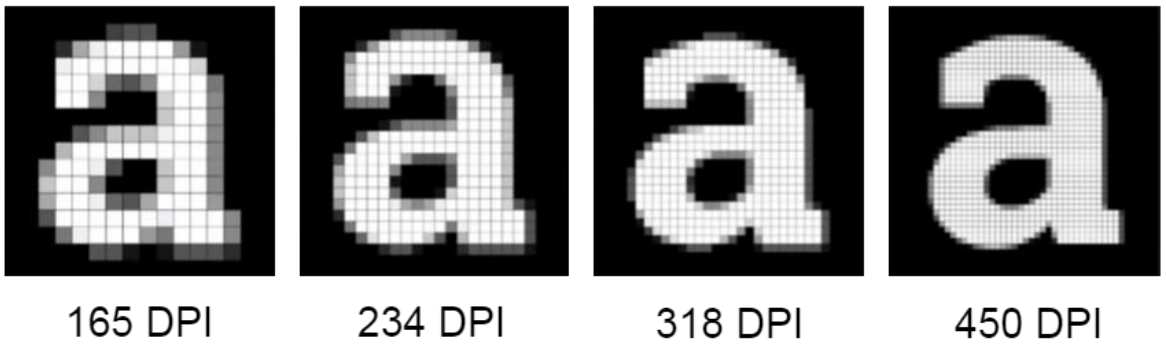


Figure 47 : Un caractère de même taille est mieux défini à plus haute résolution

Une résolution plus grande permettra soit d'afficher plus d'informations à l'écran soit le même contenu de façon plus précise. Avec des ressources adaptées, un élément prendra la même place physique mais profitera d'une meilleure qualité d'image puisque mieux définie.

Points logiques et devicePixelRatio

Sur mobile, il est indispensable de prendre en compte à la fois la définition et la résolution. En effet pour pouvoir être utilisables et distinguables, les éléments doivent rester de la même taille physique, donc avec une définition plus haute.
Pour cela existe le mécanisme de points logiques, indépendants de la définition d'un appareil. De cette façon un même élément graphique sera représenté de la même façon sur des écrans différents quelles que soient la taille physique et la définition de l'écran. Comme expliqué un élément de même taille physique sera mieux défini sur un meilleur écran.

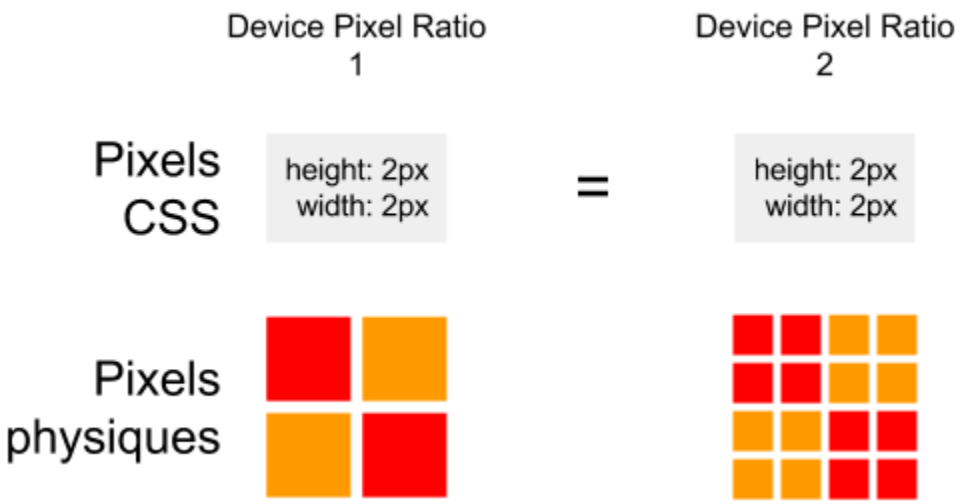


Figure 48 : Les pixels CSS sont des points logiques

En technologies web, deux mécanismes permettent d'utiliser des points logiques : les pixels CSS et le devicePixelRatio.
En effet, en CSS l'unité pixel ne représente pas les pixels physiques de l'écran mais plutôt des points logiques. Cette unité correspond souvent aux points logiques utilisée dans les APIs natives du système d'exploitation. Par exemple sur Android un pixel logique (un pixel indépendant de la résolution, ou Density-Independant Pixel) correspond à un pixel physique à une résolution de référence de 160 DPI. Sur un

appareil avec un écran de 320 DPI, le double, un point logique sera représenté par 2*2 pixels physiques.
Ce rapport entre pixels CSS, logiques, et les pixels physiques de l'écran est accessible en HTML5 via devicePixelRatio. Cette constante est accessible en JavaScript et en CSS.

Implémentation

Pour émuler les paramètres liés à l'affichage, on doit surcharger des données accessibles à la fois par l'utilisation de CSS et par l'intermédiaire de l'objet JavaScript *window*. L'émulateur prend également en compte la résolution et la définition de l'appareil émulé pour afficher l'aperçu du rendu visuel de l'application.

En CSS on se contente de calculer la taille voulue de la visualisation de l'application en pixels CSS. On applique également un redimensionnement, fonction du facteur de zoom de l'aperçu configuré par l'utilisateur via l'interface graphique. Ce facteur de zoom n'a aucune incidence sur la logique de l'application rendue, et est purement cosmétique.

```
1  document.querySelector("#EmulatorApplicationPreview").style.transform =
    "scale(" + scale + ")";
2  document.querySelector("#EmulatorApplicationPreview iframe").width =
    currentDevice.width / currentDevice.pixelratio;
3  document.querySelector("#EmulatorApplicationPreview iframe").height =
    currentDevice.height / currentDevice.pixelratio;
4  document.querySelector("#EmulatorApplicationPreview iframe").style.transform =
    "translateX(" + (-currentDevice.width / currentDevice.pixelratio / 2) + "px)";
```

Figure 49 : Code gérant la taille de la visualisation de l'application

En JavaScript, dans un navigateur Internet l'objet window[13] représente une fenêtre affichant un document DOM. Les valeurs outerWidth et outerHeight représentent la largeur et la hauteur de la fenêtre, en pixels CSS. window.pixelratio est également surchargé.
L'objet window.screen représente quant à lui l'écran sur lequel est rendue la fenêtre window. Cet objet de type Screen permet d'avoir des informations sur l'écran. Les valeurs Screen.width et Screen.height représentent donc la définition de l'écran, en pixels physiques.

[13] https://developer.mozilla.org/en-US/docs/Web/API/Window

```
1  // Override screen and window
2  var cssWidth = currentDevice.width / currentDevice.pixelratio;
3  var cssHeight = currentDevice.height / currentDevice.pixelratio;
4  override("screen", ["width", "height"], [cssWidth, cssHeight]);
5  override("window", ["outerWidth", "outerHeight", "devicePixelRatio"],
       [cssWidth, cssHeight, currentDevice.pixelratio]);
```

Figure 50 : Code surchargeant les objets window et window.screen

4.3.3 Asynchronisme et modèle d'exécution

Le modèle d'exécution des applications mobiles hybrides telles que celles développées à l'aide d'UXFME diffère de ce que l'on peut retrouver habituellement dans une application web classique dans un navigateur.

Après avoir posé les principes de la boucle d'événements et de thread de JavaScript, cette spécificité de l'asynchronisme des événements dans UXFME sera développée. Ensuite sera détaillé comment a été mis à profit l'architecture des extensions Chrome pour reproduire cette séparation de contexte d'exécution.

Le modèle d'exécution de JavaScript

Dans un contexte donné, par exemple une page web, le code JavaScript s'exécute sur un thread unique. Ce qui veut dire que le moteur JavaScript exécute les lignes de code les unes après les autres. Cela veut également dire que si un script ou une instruction prend trop de temps pour s'exécuter, c'est l'ensemble de l'application qui sera bloqué.

Néanmoins les instructions en JavaScript sont écrites pour ne pas être bloquantes, contrairement à d'autres langages. La gestion des entrées-sorties (I/O) est généralement traitée par des événements asynchrones et des fonctions de retour. Ainsi, quand l'application attend le résultat d'une requête à une base de données ou d'une requête réseau, il reste possible de traiter d'autres éléments comme par exemple les saisies utilisateur.

```
1  function a(callback) {
2      // Interrogation d'un WebService
3      // ...
4      callback(resultatServeur);
5  }
6
7  function b(result) {
8      console.log("Le résultat est : " + result);
9  }
10
11 a(b);
```

Figure 51 : Exemple de fonction de retour

Une callback ou fonction de retour est une fonction passée en argument à une autre fonction. Elle permet à cette dernière d'appeler la callback, par exemple quand un résultat est retrouvé depuis un service web, sans avoir de détails sur la fonction callback. Ce mécanisme d'inversion de contrôle est très utilisé en JavaScript.
Les appels de fonctions sont gérés grâce à une pile d'appels, qui est aussi utilisée pour les événements asynchrones.

Asynchronisme dans UXFME

Dans une application mobile hybride, il existe deux contextes d'exécution différents : la WebView qui effectue le rendu d'une page HTML5 et exécute le JavaScript, et le code dit natif qui est écrit dans le langage de développement de la plateforme mobile visée. Ainsi, non seulement les contextes d'exécution sont différents mais les technologies sous-jacentes aussi. Par exemple, il est possible sur Android en Java, par opposition au JavaScript, d'utiliser des Threads mais aussi des services du système d'exploitation.

Ces deux environnements d'exécution distincts peuvent uniquement communiquer par un système de message. Dans le cas d'UXFME, la couche JavaScript fait une requête vers une URL spécifique pour communiquer avec la couche native. Cette URL peut être de la forme *http://stz/deviceAPI/...* La partie native sera en mesure de reconnaître et de traiter ces URLs spéciales. En retour, le client UXFME appelle une fonction de retour.

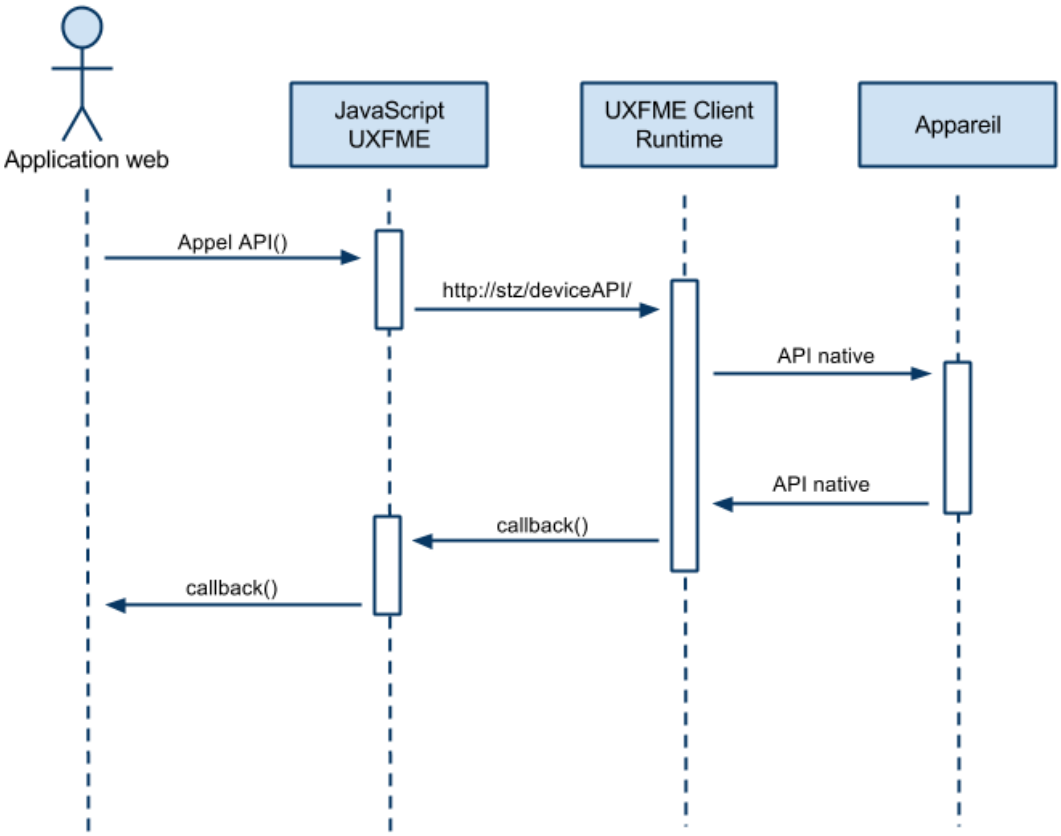


Figure 52 : Diagramme de séquence appels UXFME

Par ailleurs, il est impératif d'implémenter ce mécanisme d'asynchronisme. En effet, certaines APIs proposées par UXFME ne renvoient de résultats qu'après un laps de temps relativement long. Par exemple : la gestion de fichiers, la géolocalisation, le lecteur de code-barre, ou même l'impression. Sans asynchronisme, un appel à une de ces API UXFME bloquerait l'exécution de toute l'application.

Les extensions Chrome comme contexte d'exécution séparé

Que ce soient les scripts de la page d'arrière-plan ou les scripts de contenus, l'exécution du JavaScript d'une extension Chrome n'a jamais lieu dans le même contexte que la page. Pour l'émulateur, cela veut dire que l'extension se comporte comme la partie native d'UXFME en s'exécutant en parallèle du contenu web de l'application mobile hybride.

Cette spécificité a été exploitée dans l'implémentation de l'émulateur pour mettre en place une séparation de contexte d'exécution et garantir l'asynchronisme entre l'application web et le client UXFME, ici émulé.

```
deviceapis.device.getProperties(displayDeviceProperties);

function displayDeviceProperties(deviceProperties) {
    alert("Brand: " + deviceProperties.brand +
          " / Model: " + deviceProperties.model +
          " / OS: " + deviceProperties.os +
          " / Screen width: " + deviceProperties.screenWidth +
          " / Screen height: " + deviceProperties.screenHeight);
}
```

Figure 53 : API UXFME avec une fonction de retour

La bibliothèque JavaScript UXFME expose les APIs clients et fait appel à la partie native à l'aide d'URLs spéciales. Selon les plateformes, ces URLs débutent par *stz://deviceAPI/* ou par *http://stz/deviceAPI*. Comme la partie native du client UXFME, l'API chrome.webRequest des extensions Chrome permet d'intercepter les requêtes réseaux quelles qu'elles soient : navigation, requêtes HTTP asynchrones, chargement de ressources. Il suffit donc d'intercepter ces URLs spéciales et de décoder les paramètres présents dans l'URL utile à l'exécution de l'API client émulée, comme le ferait la partie native du client UXFME. Garder ce même mécanisme d'URL spéciales permet à l'émulateur d'être compatible avec la bibliothèque JavaScript UXFME sans modifications de cette dernière.

```
http://stz/deviceAPI/DEVICE?action=getProperties&callback=deviceapis.device.getPropertiesCallback
```

Figure 54 : Appel interne à une device API

Dans l'exemple de la fonction deviceapis.device.getProperties(), l'émulateur doit retourner des propriétés de l'appareil mobile via la fonction de retour. Ces données, telles que la marque et le modèle, ou la taille de l'écran, sont extraites de la base de données d'appareils.

```
// Code injection
function inject(callbackSignature)
{
    var s = document.createElement("script");
    s.textContent = callbackSignature;
    document.documentElement.appendChild(s);
}
```

Figure 55 : Injection de JavaScript dans la page de l'application émulée

A la fin du traitement, l'émulateur doit appeler la fonction de retour JavaScript pour transmettre les résultats. L'émulateur étant dans un contexte d'exécution différent, il existe une solution simple : l'injection de JavaScript via le DOM. Le principe est simple : si on ne peut pas exécuter de code dans le contexte de la page, on peut néanmoins insérer une balise HTML script contenant le code à exécuter.

4.4 Interface utilisateur

De par la conception de l'émulateur UXFME, l'interface utilisateur se doit d'être non seulement ergonomique et intuitive mais aussi modulaire.
En outre l'interface de l'émulateur UXFME ne doit en aucun cas interférer avec l'usage de l'application émulée elle-même, et ne doit donc pas altérer la capacité à tester l'intégralité du rendu et des interactions qui y sont rendus possibles.
Chaque bloc de l'interface représente des réglages ou informations liées à une fonctionnalité. Après une vue d'ensemble de l'interface réalisée, chacun de ces blocs seront détaillés.

4.4.1 Vue d'ensemble

L'utilisateur peut ouvrir une page web ou une application dans Chrome via la barre de navigation. L'émulateur UXFME est débrayable à tout moment grâce à l'utilisation d'un bouton d'action, qui indique par ailleurs si l'émulation est active ou non.

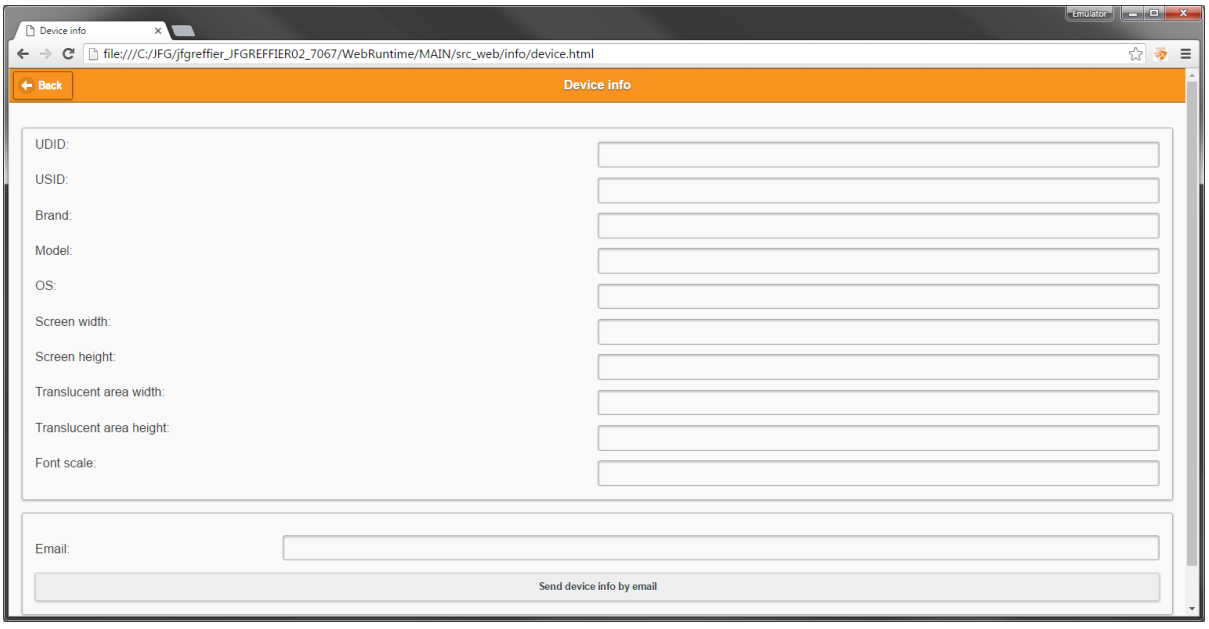


Figure 56 : Google Chrome affichant une application UXFME

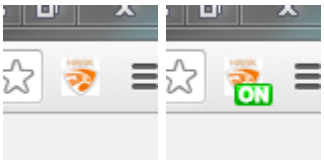


Figure 57 : Bouton d'action de l'émulateur UXFME désactivé, puis activé

A l'activation, la page est rechargée et l'interface utilisateur de l'émulateur apparaît. L'affichage du site web ou de l'application est adapté à l'appareil mobile et les APIs UXFME sont utilisables. Par exemple, les informations concernant l'appareil (marque, modèle, version du système d'exploitation) peuvent être retrouvées et affichées dans l'application mobile hybride.

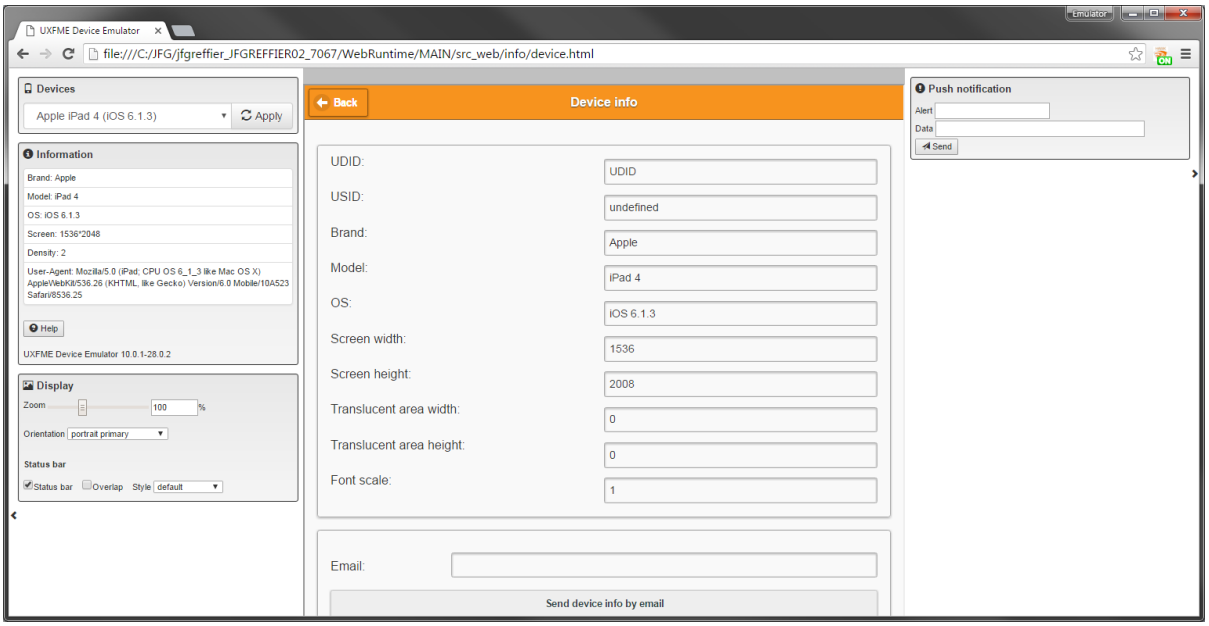


Figure 58 : Emulateur activé avec les API disponibles

L'interface utilisateur est composée des éléments suivants :

- sélection des appareils depuis la liste extraite de la base de données des appareils issue de la Device Certification.
- panneau d'information affichant les données pertinentes sur l'appareil émulé.
- réglages de l'affichage permettant d'appliquer un zoom sur la visualisation de l'application, de spécifier l'orientation de l'affichage et d'agir sur la barre de statut.
- notification push pour simuler l'envoi d'une notification à l'application.

La visualisation de l'application mobile hybride émulée est présentée au centre.

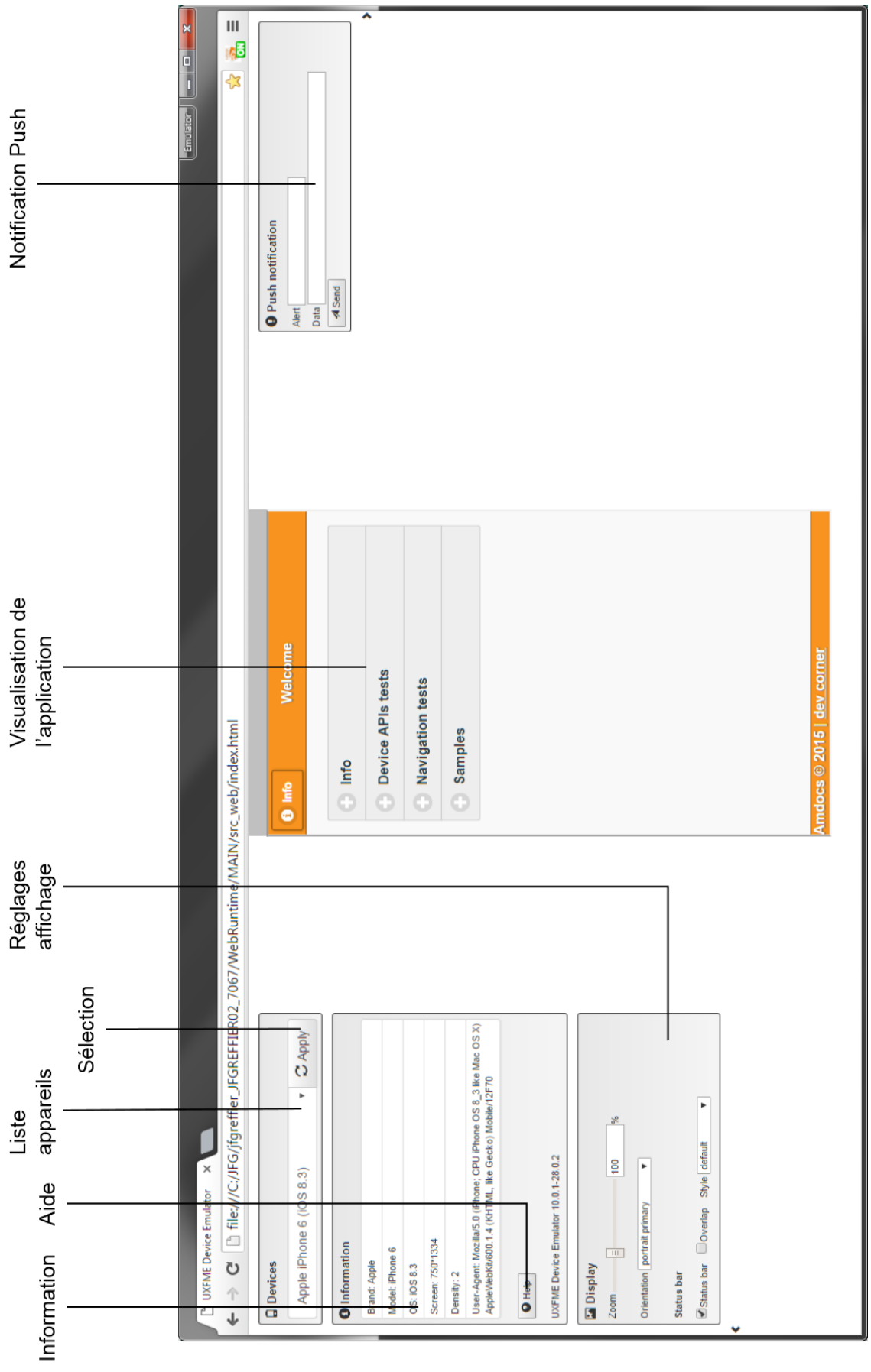


Figure 59 : Vue d'ensemble

4.4.2 Sélectionner un appareil

Ce panneau permet de choisir l'appareil à émuler parmi la liste issue de la base de données d'appareils. Une fois le choix validé, le contenu de l'application émulée est rechargé afin de mettre à jour l'affichage et le comportement de l'application. Le panneau d'information est également mis à jour.

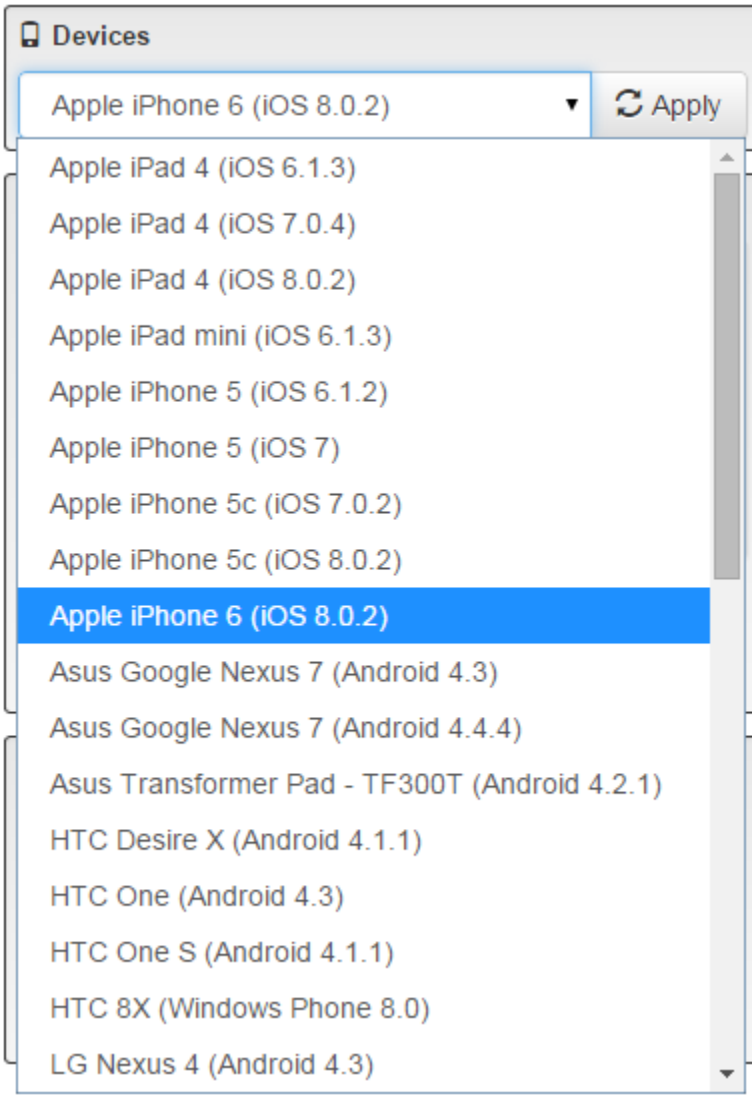


Figure 60 : Sélection d'appareil à émuler

4.4.3 Panneau Information

Ce panneau liste les informations utiles concernant l'appareil émulé :

- la marque
- le modèle
- le système d'exploitation et sa version
- la définition de l'écran, en pixels
- le ratio pixels logiques-pixels physiques, ou devicePixelRatio
- la chaîne de User-Agent

Un bouton "Help" permet d'accéder au guide d'utilisation de l'émulateur sous la forme d'un document PDF. Enfin, la version de l'émulateur UXFME est indiquée.

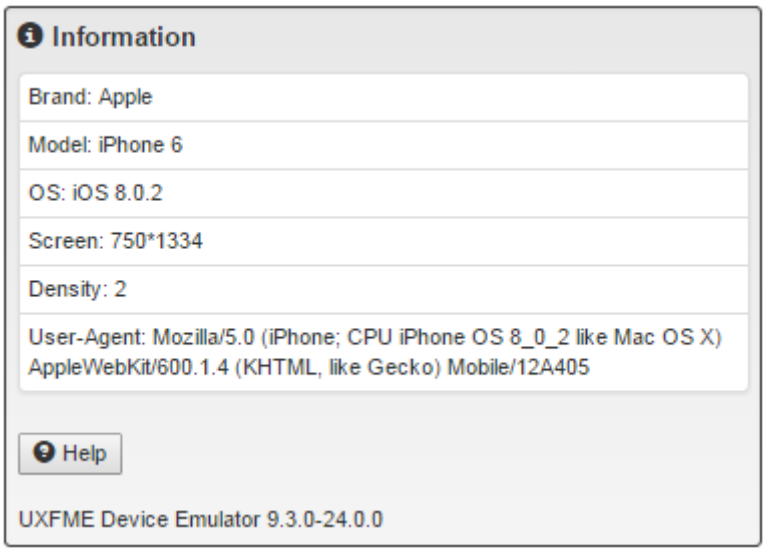


Figure 61 : Informations de l'iPhone 6 iOS 8.0.2

4.4.4 Panneau affichage

Le premier réglage permet d'appliquer un facteur de zoom à la visualisation de l'application émulée. Le zoom n'est là que faciliter l'aperçu de l'application à l'utilisateur de l'émulateur UXFME, par exemple pour les appareils ayant une très grande définition.

Le deuxième réglage permet de sélectionner une orientation parmi une liste :

- portrait primaire. Appareil en portrait, dans sa position naturelle. Par exemple pour un smartphone iPhone, le bouton physique (Home) est situé en bas.

- portrait secondaire. Portrait mais à 180° de sa position habituelle.
- paysage primaire. Appareil en paysage, dans sa position naturelle. Par exemple pour une tablette iPad, le bouton physique (Home) est alors situé en bas. Pour un téléphone, cette position est généralement le paysage gauche.
- paysage secondaire. Paysage mais à 180° de sa position habituelle.

Il est à noter que ce bloc interagit avec les APIs clientes UXFME deviceapis.webview.layout et deviceapis.webview.screenorientation. En effet, les fonctions de retour préalablement enregistrées grâce à l'API UXFME seront notifiées d'un changement d'orientation. Inversement, si l'affichage est modifié programmatiquement l'orientation affichée par l'émulateur ainsi que la visualisation de l'application seront mis à jour en conséquence.

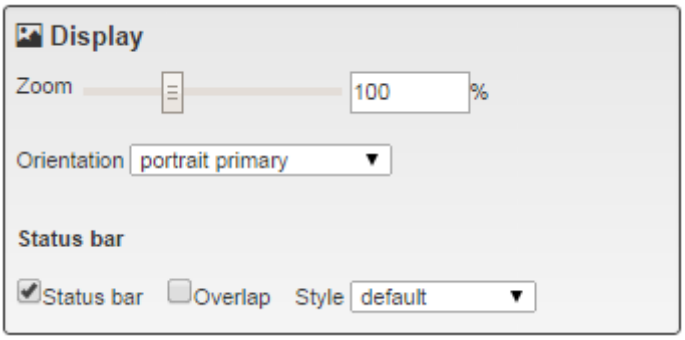


Figure 62 : Panneau affichage

La dernière partie permet d'afficher ou de cacher la barre de statut de l'appareil, affectant ainsi potentiellement l'espace disponible sur l'écran pour le rendu visible de l'application. L'option "overlap" permet, si le système d'exploitation le supporte, d'afficher la barre de statut par-dessus le contenu de l'application émulée. Le style de la barre de statut pourra être sur fond noir, sur fond blanc ou partiellement transparente.
De la même façon que l'orientation, ce réglage est lié à l'API client UXFME deviceapis.webview.layout. Cette API permet de changer les attributs de la barre de statut, ce qui sera reflété dans l'interface utilisateur de l'émulateur.

4.4.5 Notification Push

Cette interface permet d'émuler l'envoi d'une notification push à l'application. Pour cela, l'application doit avoir enregistré une fonction de retour à l'aide de l'API deviceapis.notification qui lui permettra de réagir en fonction des données transmises à l'application par le biais de la notification push.

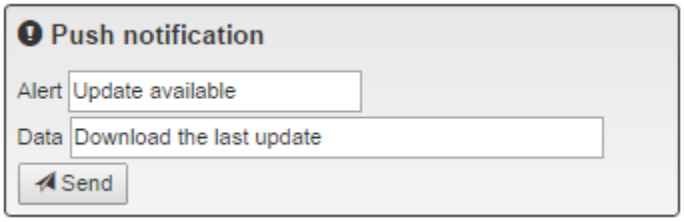


Figure 63 : Edition d'une Notification Push

Pour envoyer une notification push :

- éditer le champ "Alert" qui correspond au titre de la notification push.
- éditer le champ "Data" qui correspond aux données qui seront transmises à l'application via la notification push.
- cliquer sur "Send". Une notification push apparaît dans le bas de l'écran. En cliquant dessus, comme l'utilisateur d'un appareil mobile le ferait au travers du panneau de notification, la ou les fonctions de retours préalablement enregistrées par l'application sont appelées.

La réalisation de l'émulateur UXFME a permis dès les premières itérations de livrer un produit fonctionnel. Tout ce qui a été conçu n'a cependant pas été réalisé, et l'application devra évoluer et être maintenue depuis cette version initiale. En complément de ces aspects, la partie suivante dresse un bilan personnel.

5 Bilan

Cette dernière partie vise à établir un bilan du projet. Tout d'abord en détaillant ce qui reste à faire et les difficultés rencontrées. Ensuite, en présentant la maintenance et les perspectives d'évolution de l'application. Enfin, cette partie se termine par un bilan personnel.

5.1 Résultats

5.1.1 Reste à faire

En scrum, les fonctionnalités qui restent encore à concevoir et à réaliser sont présentes sous forme de stories dans le backlog produit. En début de sprint sont sélectionnées les stories qui pourront être réalisées, en fonction des priorités et de la capacité de l'équipe. C'est le Product Owner, dans l'intérêt des parties prenantes et en accord avec elles, qui définit les priorités et les stories du backlog produit.
Par ailleurs, les récits utilisateurs du produit peuvent être plus ou moins détaillés. On affine alors la story avant de l'ajouter dans un sprint de développement. Les stories listées ci-après n'ont pas été encore réalisées car jugées moins prioritaires. De plus, le travail d'affinage n'a pas été fait et elles n'étaient donc pas rédigées. On peut donc proposer dans ce document les récits utilisateurs suivants.

Géolocalisation

Il doit être possible de configurer toutes les valeurs accessibles depuis l'API UXFME de géolocalisation par une interface graphique claire. Ces valeurs doivent être accessibles depuis l'application UXFME dans l'émulateur.

Le concept de l'interface utilisateur de la fonction est sous forme de schéma en fil de fer “wireframe”.

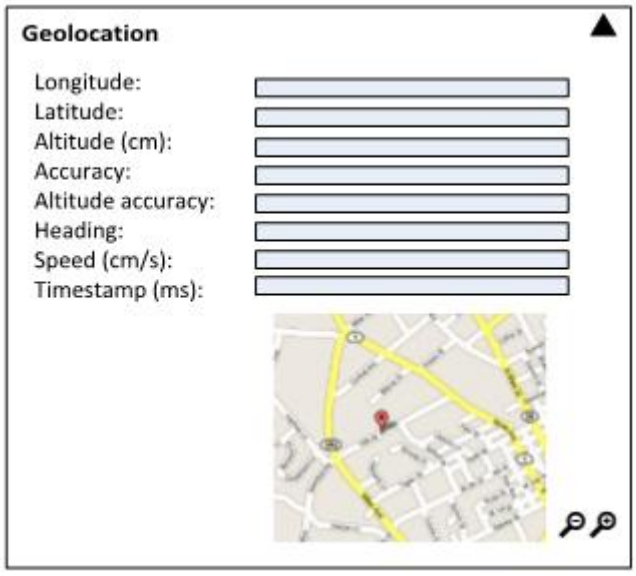


Figure 64 : Schématisation en fil de fer de la fonction géolocalisation

Accéléromètre

Il doit être possible de configurer facilement les valeurs x, y et z d'accéléromètres émulés, accessibles par une API dans une application UXFME.

Orientation

Il doit être possible de configurer facilement les valeurs alpha, beta et gamma d'un capteur émulé, accessibles par une API dans une application UXFME.

Enfin la story "Mise à jour d'application" n'a pas encore été réalisée étant donné sa faible priorité et sa complexité. En effet, la fonctionnalité UXFME de mise à jour et de mise à disposition d'installateur pour appareil mobile est fortement dépendante de l'environnement des appareils mobiles et de serveur distant du UXFME Applications Manager.

Mise à jour d'application

Il doit être possible de gérer le cycle de vie de l'application émulée, en spécifiant une instance du UXFME Applications Manager qui sera la source de mises-à-jour, afin de pouvoir en tester les effets.

5.1.2 Difficultés rencontrées

Comme évoqué précédemment, la conception et la réalisation initiale de l'émulateur UXFME se sont déroulées du drop 19 au drop 21. Le nombre de fonctionnalités livrées à l'issue de la conception et de la réalisation initiale n'est pas en accord avec le plan initial, et ce pour plusieurs raisons.

Tout d'abord la phase d'exploration technique (spike) a été beaucoup plus longue que prévue. En effet, il a fallu explorer et tester l'ensemble des possibilités d'extension de Chrome DevTools et de son mode d'émulation mobile avant d'écarter définitivement cette solution. Les possibilités d'extensions via l'API officielle, de modification de Chrome ou la faisabilité d'un outil externe ont été explorées dans le détail, avec la réalisation de plusieurs prototypes.

Ensuite, plusieurs tâches de réalisation, durant le drop 20, étaient manifestement sous-évaluées. Par exemple l'estimation originale supposait trois heures pour l'intégration de l'aide web, au final plusieurs jours ont été nécessaires. Cette tâche consistait à intégrer une aide en ligne produite par le logiciel Adobe RoboHelp.

Figure 65 : Logo de RoboHelp

Adobe RoboHelp est un logiciel de création et de publication de contenu d'aide. Il permet d'exporter vers de nombreux formats : PDF, XHTML, word, livre numérique epub…
Après investigations, il s'est avéré que l'aide générée par le logiciel n'était pas compatible avec les règles de sécurité de Chrome. Par ailleurs, le problème est connu d'Adobe qui propose un patch pour RoboHelp 10.0.1[14]

[14] https://helpx.adobe.com/robohelp/kb/webhelp-output-fails-load-chrome.html

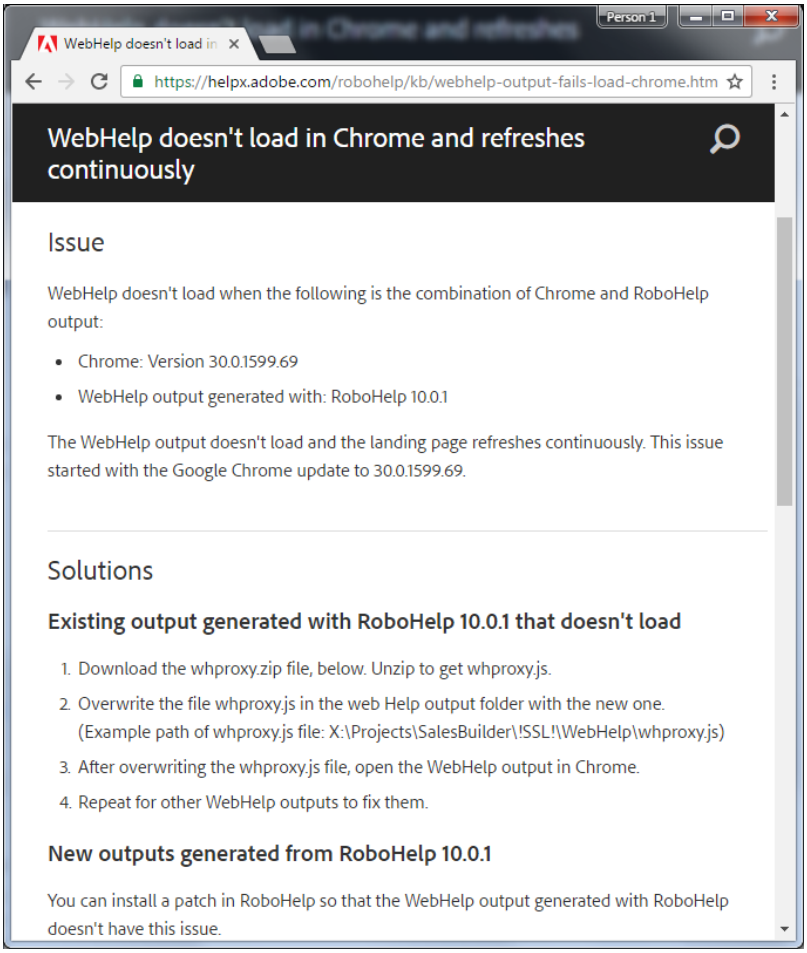


Figure 66 : Problème documenté par Adobe

Amdocs utilisant une version plus ancienne, il a été proposé une résolution en deux temps : ajouter l'aide au format PDF, et faire une demande pour mettre à jour RoboHelp pour ensuite pouvoir appliquer le patch fourni.

Le drop 21 a été une période principalement consacrée à la résolution de bogues et plus généralement à la prise en compte des retours utilisateurs. En effet, le drop 20 constituait la première livraison publique de l'émulateur UXFME.

5.2 Évolution

5.2.1 Maintenance

Comme d'autres produits constituants le framework UXFME, l'émulateur est maintenu durant tout son cycle de vie de plusieurs façons.

La norme ISO/IEC 14764 définit quatre types de maintenances :

- maintenance perfective
- maintenance préventive
- maintenance adaptative
- maintenance corrective

La maintenance perfective a pour but principal d'améliorer la qualité du code et de l'architecture. La maintenance préventive a pour but d'améliorer le produit en anticipant les problèmes.
La maintenance adaptative quant à elle permet d'adapter le produit aux changements d'environnement. Enfin, la maintenance corrective permet de résoudre les bogues.

Maintenance perfective et préventive

Pour garantir le niveau de qualité pendant le développement d'un logiciel, la pratique recommandée de Scrum est d'avoir recours à la revue par un pair. Chaque tâche effectuée pendant la réalisation d'une story est mise en état "pour revue". Un autre membre de l'équipe procède alors à une revue de la réalisation et fait part de ses suggestions, le contributeur original corrigeant ensuite. Les remarques peuvent concerner le style du code, l'orthographe, les commentaires, ou être une remarque d'ordre plus générale. Cela permet d'avoir une première validation et un nouveau point de vue sur la solution technique développée.
Ensuite, et pour faciliter le développement des itérations futures, il a été procédé à une revue d'architecture de l'existant après quelques Drops. En effet, de par la nature itérative des projets développés en Scrum il est nécessaire de vérifier que l'architecture est toujours adaptée et être pro-actif sur d'éventuels futurs bogues.
Enfin, l'équipe UXFME procède à une phase de stabilisation à chaque fin de Drop. C'est l'occasion de procéder à des tests de régression sur les éléments constituants

la solution. Les tests de régression consistent à vérifier les scénarios de tests reflétant les cas d'utilisations les plus courants. Le fait de faire ces tests régulièrement permet de repérer toute régression dans le comportement de l'application.

Maintenance adaptative

L'émulateur UXFME doit être adapté suite aux changements d'environnements.
Tout d'abord, il faut s'aligner sur les changements de fonctionnalités ou de signatures d'API JavaScript UXFME à chaque Drop.
La base de données des appareils certifiés est enrichie en général de façon annuelle. En effet, des appareils mobiles sont choisis pour être certifiés à chaque nouvelle version de la suite Amdocs Customer Experience Solution. Ces appareils sont sélectionnés pour être représentatifs du marché des appareils mobiles.

Maintenance corrective

Les bogues sont découverts et remontés de trois façons : soit durant une phase de stabilisation où des tests ont montré un défaut, soit en interne pendant l'utilisation de l'émulateur par l'équipe UXFME, soit par un client interne ou externe.
Dans tous les cas, le défaut est rentré dans un outil de suivi de bogues (Mantis bug tracker) avec une description la plus précise possible ainsi que les informations complètes sur les étapes nécessaires pour le reproduire. Le bogue est alors catégorisé puis trié par la Product Owner. C'est-à-dire qu'une priorité est allouée, et le défaut peut être assigné à un développeur qui sera alors en charge des investigations afin de résoudre le bogue.

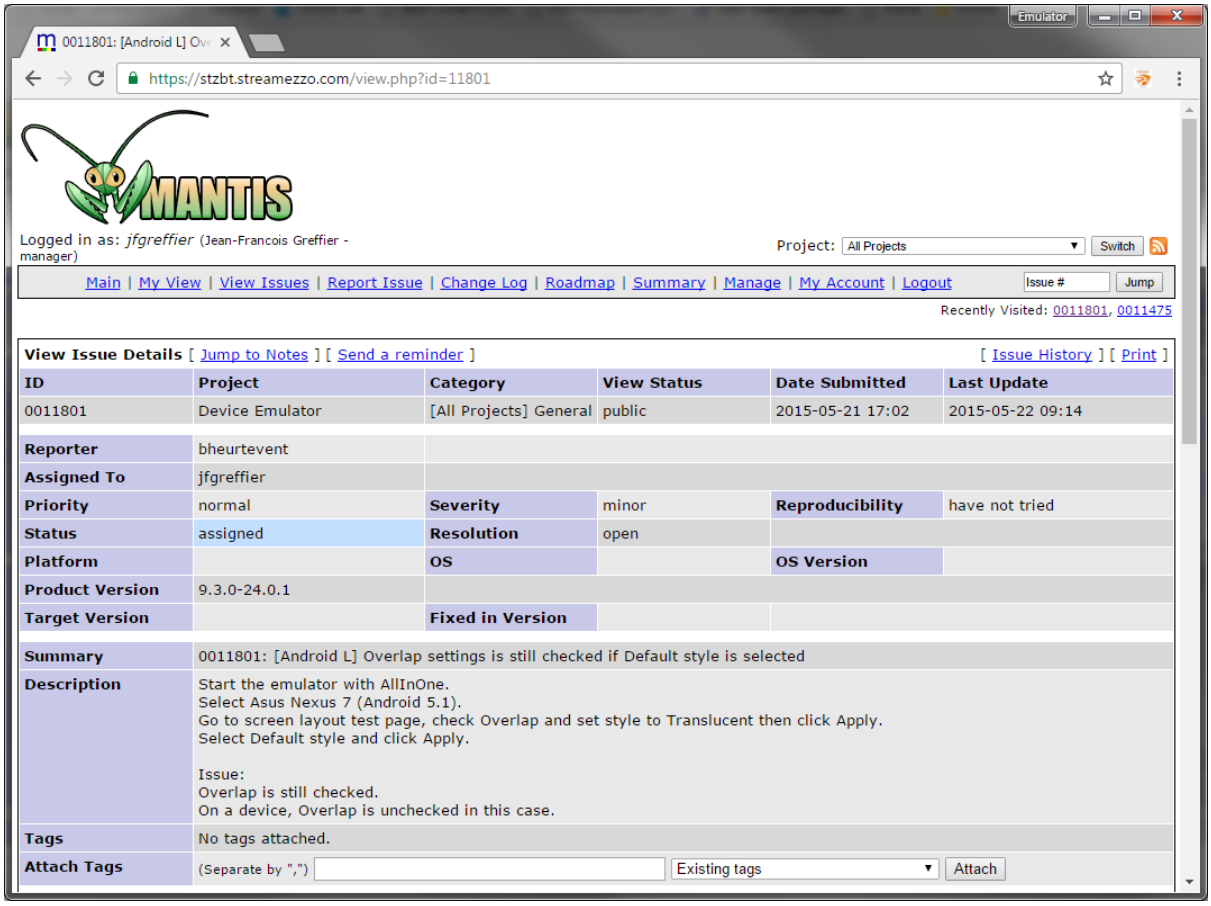


Figure 67 : Rapport de bogue de l'émulateur UXFME dans Mantis

5.2.2 Perspectives

Au moins deux stories étaient anticipées dans le backlog produit pour des évolutions futures. Elles ont en commun de proposer des points de configuration supplémentaires à l'utilisateur quant aux données accessibles depuis un appareil mobile émulé.

Configuration des données d'appareil émulé

Il doit être possible de configurer l'ensemble des données accessibles depuis l'émulation d'un appareil mobile :

- identification de l'appareil accessible depuis les APIs UXFME
- liste des contacts
- liste des événements de l'agenda
- lien du système de fichier local de l'ordinateur vers l'appareil émulé

Configuration des interactions

Il doit être possible de configurer les données renvoyées par les usages suivants :

- géolocalisation
- accéléromètre
- orientation
- informations réseau
- lecteur code-barres

Par ailleurs, pour prévoir les évolutions futures du produit, il semble important de faire un travail de veille technologique. Par exemple Chrome DevTools et ses outils mobiles se sont révélés très intéressants même s'ils ne satisfaisaient pas les besoins de possibilités d'extensions nécessaires au projet. L'équipe UXFME procède donc régulièrement à de la veille pour vérifier si cette situation change. Plus généralement, il est important de vérifier s'il y a de nouvelles solutions émergentes et comment évoluent les projets précédemment étudiés, comme Apache Ripple.

5.3 Bilan personnel

D'un point de vue personnel, j'ai trouvé l'ensemble de cette expérience très enrichissante.

Tout d'abord, la rédaction de ce mémoire d'ingénieur a été l'occasion de prendre du recul sur mon activité professionnelle, le fonctionnement de l'équipe où je travaille, sur le fonctionnement d'Amdocs, et le produit UXFME. La rédaction d'un document étayé, documenté, concis et clair constituait un exercice en soit.
Plus généralement ma démarche au sein de l'EICNAM m'a permis de renforcer mes compétences et d'en acquérir de nouvelles, accompagnant mon parcours professionnel.

Durant la partie étude de ce projet j'ai pu participer à l'exploration technique et à la conception de l'architecture. Ce projet m'a permis de découvrir de nouvelles technologies grâce à cette phase d'exploration technique importante. Les choix techniques ayant conduit à créer de toutes pièces cet émulateur m'ont permis de renforcer mon expertise du développement mobile natif et web.
Enfin, pouvoir suivre un projet de sa conception à son implémentation est une expérience enrichissante.

6 Conclusion

Les technologies mobiles n'ont sans doute pas fini de révolutionner l'informatique personnelle, possiblement en évoluant vers les objets connectés et l'informatique ubiquitaire. Les applications mobiles hybrides représentent une excellente solution pour unifier les mondes des smartphones, des tablettes et des ordinateurs personnels.

Ce projet d'émulateur est une nouvelle brique logicielle s'ajoutant à la solution UXFME, facilitant le développement d'applications mobiles au sein d'Amdocs et pour ses clients à travers le monde.

A l'aide de Scrum, l'équipe a été capable de répondre avec une grande réactivité à des besoins qui n'étaient pas toujours explicités. Il a ensuite été possible d'itérer et donc de raffiner à la fois l'expression des besoins et le logiciel résultant.

Il y avait une grande latitude quant aux choix technologiques et architecturaux de la solution. Il a donc été possible d'être véritablement force de proposition, mettant ainsi en avant l'expertise de l'équipe UXFME dans le domaine du mobile. Cette expertise, forte de nombreuses années dans le milieu mobile, permet d'apporter une compréhension des spécificités du mobile du point de vue technique, mais aussi en termes d'usabilité, et de connaissance du marché.

A titre personnel, j'ai eu la chance d'être le contributeur principal à ce projet d'émulateur pour framework d'applications mobiles hybrides, que ce soit dans la partie étude ou technique.

L'expérience accumulée pendant ce projet permet d'imaginer d'autres pistes pour améliorer l'expérience utilisateur des développeurs utilisant UXFME pour créer des applications mobiles hybrides. On peut par exemple imaginer un émulateur distant, en logiciel en tant que service, mais aussi des solutions permettant d'améliorer les outils de débogage à distance sur mobile.

7 Annexes

7.1 Annexe 1 : Communiqué de presse médiamétrie

Le communiqué de presse de Médiamétrie “Les chiffres clés de la mesure web analytics de Médiamétrie” de Septembre 2015 n’étant plus disponible en ligne, il est reproduit dans cet annexe.

Ce document montre à la fois qu’en France le trafic web se fait majoritairement depuis un appareil mobile, et que les systèmes d’exploitation iOS et Android sont les plus présents.

COMMUNIQUÉ DE PRESSE

Levallois, le 9 octobre 2015

Mediametrie

eStat'Web

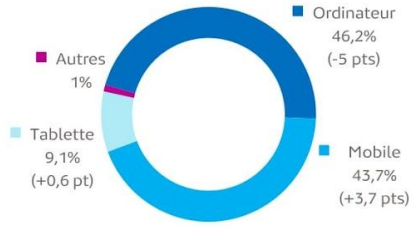
Les chiffres clés de la mesure web analytics de Médiamétrie
Septembre 2015

Les résultats sont fondés sur les performances des clients souscripteurs à la solution eStat'Web de Médiamétrie, qui mesure la fréquentation des sites web.

Plus de la moitié des visites réalisées sur écrans mobiles

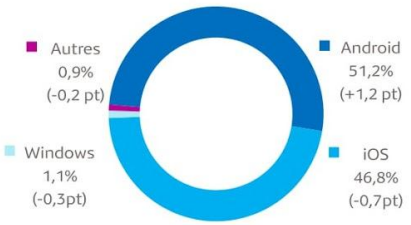
Médiamétrie publie un bilan sur la répartition par terminal des visites des sites mesurés par eStat ainsi qu'une répartition des systèmes d'exploitation mobiles et des navigateurs au mois de septembre 2015.

Répartition des visites par type de terminal



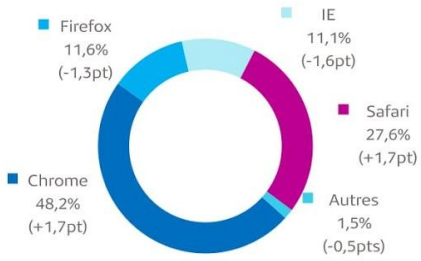
En septembre 2015, plus de 4 visites de sites sur 10 (43,7%) se sont faites depuis un téléphone mobile soit une progression de 3,7 points par rapport au mois de juin. La tablette, également en progression (+0,6 point) atteint plus de 9% des visites. A contrario, l'utilisation de l'ordinateur pour se connecter à internet est en baisse et représente moins d'1 visite sur 2. A noter que les visites de site depuis les télévisions et consoles de jeux atteignent pour la 1ère fois 1% du trafic.

Répartition des visites par système d'exploitation mobile



Le système d'exploitation mobile Android est le seul à enregistrer une hausse de sa part de visites au mois septembre par rapport au mois de juin (+1,2 point). Ainsi, ce sont 51,2% des visites des sites qui ont été réalisées depuis un terminal Android contre 46,8% pour iOS.

Répartition des visites par navigateur Internet



Tous terminaux confondus le navigateur Google Chrome confirme sa position dominante avec près d'1 visite sur 2 (48,2%). Safari (et sa version mobile) continue sa progression et maintient sa 2ème place avec 27,6% des visites.
Respectivement 3ème et 4ème des navigateurs les plus utilisés, Firefox et Internet Explorer, malgré une légère baisse, représentent chacun plus de 11% des visites des sites.

COMMUNIQUÉ DE PRESSE

eStat'Web

Fréquentation des sites internet : Top 10 des progressions du mois

A la rentrée, les sites du Groupe CERISE **Gentside** et **Ohmymag** occupent les deux premières places du classement des progressions avec respectivement 2,7 et 2,5 millions de visites supplémentaires par rapport au mois d'août. **RTL.fr** vient compléter ce podium avec une hausse de sa fréquentation de près de 2,3 millions de visites en un mois . Les autres stations du groupe enregistrent elles aussi une augmentation de visites significatives avec +38% pour **RTL2.fr** (+ 399 000 visites) et +19% pour le site **FUNRADIO.fr** (+364 000 visites).

En 4ème position, le site **Les Inrockuptibles** enregistre une fréquentation en hausse de 1,8 million de visites (+37%), tout comme **Studyrama.com** qui gagne lui aussi en un mois plus de 1 millions de visites (+56%).

Enfin, tout comme le site d'annonces **pap.fr** (+ 888 000 visites), **Normandie Actu** (+516 000 visites) et **Lagazettedescommunes.com** (+425 000 visites) profitent eux aussi de la fin des vacances pour débuter une nouvelle année scolaire avec un trafic en hausse.

Classement des plus fortes progressions du mois de septembre en nombre de visites

Rang	Sites	Visites sept 2015 (mois normé)	Évolution / août 2015 (valeur)	Évolution / août 2015 (%)
1	Gentside (Groupe CERISE)	27 827 664	+2 704 506	+11%
2	Ohmymag (Groupe CERISE)	32 067 778	+2 453 360	+8%
3	RTL.fr	14 102 324	+2 282 690	+19%
4	Les Inrockuptibles (Groupe Les Inrockuptibles)	6 592 745	+1 770 889	+37%
5	Studyrama.com (Groupe Studyrama)	2 830 609	+1 011 512	+56%
6	pap.fr (Groupe PAP)	7 209 309	+888 201	+14%
7	Normandie Actu	2 692 727	+516 320	+24%
8	Lagazettedescommunes.com (Groupe Moniteur)	1 102 398	+425 429	+63%
9	RTL2.fr	1 440 920	+398 688	+38%
10	FUNRADIO.fr	2 274 655	+364 094	+19%

Définitions :
Visite de site : consultation d'un site, c'est-à-dire chargement par l'internaute d'au moins une page d'un site web. Plusieurs visites peuvent correspondre au même visiteur.
Mois normé : mois théorique composé d'un nombre de jours et de week-ends fixe (30,4375 jours, soit 2/7ème de jours de week-ends pour 5/7ème de jours de semaine). Utilisé pour la comparaison des visites mois par mois, il compense les variations du nombre de jours et de week-ends des mois existants.

COMMUNIQUÉ DE PRESSE
eStat'Web

A propos de Médiamétrie

Leader des études médias, Médiamétrie observe, mesure et analyse les comportements du public et les tendances du marché. Créée en 1985, Médiamétrie développe ses activités en France et à l'international sur la Télévision, la Radio, l'Internet, le Cinéma, le Téléphone Mobile et le Cross-Médias. En 2014, Médiamétrie a réalisé un chiffre d'affaires de 82,4 M€.
Avec les solutions eStat, Médiamétrie est la référence en France de la mesure des contenus numériques fondée sur une technologie de marqueurs en « site centric » dédiés aux besoins des éditeurs et régies en matière de pilotage et d'optimisation de leurs contenus web, mobile et le développement de leurs applications (mobiles / tablettes / TV connectées...). Chaque client dispose d'une interface dédiée couvrant plus d'une centaine d'indicateurs et d'analyses, comme le trafic, le comportement de l'internaute, la navigation, la provenance, la géolocalisation, etc.

Web : www.mediametrie.fr
Twitter : www.twitter.com/Mediametrie
Facebook : www.facebook.com/Mediametrie

Contact Presse :
Isabelle Lellouche Filliau
Tél : 01 47 58 97 26
ilellouche-filliau@mediametrie.fr

7.2 Annexe 2 : Récits utilisateurs

Cet annexe liste les stories qui ont été réalisées pendant la durée du projet couverte dans ce mémoire. Les récits utilisateurs apparaissent tels que dans l'outil de suivi de projet Atlassian Jira, donc en anglais. Elles ont été écrites par le Product Manager.

Spike UXFME-1071 UXF Web IDE / UXFME Emulation

Acceptance criteria:
1) Specify clearly what are the expected features for UXFME in-browser device emulation
2) Identify what can be reused to build it
3) Identify what will need to be developed to cover the all picture, with related LoE estimation

Story UXFME-2402 Device Emulator / Devices selection

It must be possible to select among the list of certified devices to tune the device emulator:
- user agent (incl. brand / model + OS)
- display: screen dimensions + device pixel ratio

Story UXFME-2403 Device Emulator / Device data configuration

it must be possible to configure the simulated data on the device:
- device ids
- list of contacts
- list of events (calendar)
- root for device file system on local file system

Story UXFME-2405 Device Emulator / Device display configuration

it must be possible to configure the simulated display of the device:
- screen orientation (switch from possible options)
- status bar display (incl. overlap, network indicator)

Story UXFME-2406 Device Emulator / Application update

It must be possible to trigger application update, i.e. specify Applications Manager instance and detect / download version from there.

Spike UXFME-2438 Device Emulation / Chrome extension packaging

It must be possible to install and execute UXFME Device Emulator Chrome extension in standard Chrome?
Google policy has recently changed regarding non-Chrome Web Store extensions which are now disabled, see
http://chrome.blogspot.fr/2014/05/protecting-chrome-users-from-malicious.html.

Options:
- Local installation (?)
- Installation via Enterprise policy (?)
- Distribution though Google Web Store

Story UXFME-2443 Device Emulator / Push notification emulation

It must be possible from the Device Emulator extension to:
1) forge notification
2) send notification

The purpose of this feature is to test locally how the application reacts to the reception of a push notification.
It is somewhat comparable to what we are already offering on devices, by accessing test web page in device web browser to trigger the wake-up of application, specifying which data it is supposed to receive.

Story UXFME-2819 Device Emulator / Implementation review and documentation

UXFME Emulator high level design figure must be provided for implementation review
Sub tasks :
The high level figure should describe the different components, their execution contexts and how they communicate.
The device list is currently made of a JSON file. The structure of it should be reviewed and documented.
Should include new features documentation.

7.3 Annexe 3 : Story UXFME-2443 Push notification emulation

Cet annexe détaille le récit utilisateur et les tâches concernant l'émulation de la notification push.

On pourra noter que la story est constituée de plusieurs tâches, dont une de test :

- UXFME-2533 UI/ECP: input for push notification data + submit button
- UXFME-2534 trap push notification registration and unregistration (callback)
- UXFME-2535 inject data upon push notification submission
- UXFME-2728 QA Testing / Local push notification
- UXFME-2825 Forge UXFME push notification

[UXFME-2443] Device Emulator / Push notification emulation	
Status:	Resolved
Project:	UXFME
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	Story	**Priority:**	10
Reporter:	Christophe Michel	**Assignee:**	Christophe Michel
Resolution:	Fixed	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	0.5h	**Remaining Estimate:**	Not Specified
Σ Time Spent:	18.5h	**Time Spent:**	Not Specified
Σ Original Estimate:	19h	**Original Estimate:**	Not Specified

Sub-Tasks:

Key	Summary	Type	Status	Assignee
UXFME-2533	UI/ECP: input for push notification d...	Technical task	Resolved	Jean-Francois Greffier
UXFME-2534	trap push notification	Technical task	Resolved	Jean-Francois Greffier

		registration a...			
	UXFME-2535	inject data upon push notification su...	Technical task	Resolved	Jean-Francois Greffier
	UXFME-2728	QA Testing / Local push notification	Technical task	Resolved	Christophe Michel
	UXFME-2825	Forge UXFME push notification	Technical task	Resolved	Jean-Francois Greffier
Story Points:	8				
Epic Link:	Device Emulation				
Sprint:	Drop 20 Dev Sprint 1, Drop 20 Dev Sprint 2, Drop 20 Dev Sprint 3, Drop 20 Dev Sprint 4				

Description

It must be possible from the Device Emulator extension to:
1) forge notification
2) send notification

The purpose of this feature is to test locally how the application reacts to the reception of a push notification.
It is somewhat comparable to what we are already offering on devices, by accessing test web page in device web browser to trigger the wake-up of application, specifying which data it is supposed to receive.

Generated by Jean-Francois Greffier using JIRA 6.3.9#6339-sha1:46fa26140bf81c66e10e6f784903d4bfb1a521ae.

8 Bibliographie

[1] Collectif, 2001. *Manifeste pour le développement Agile de logiciels*. http://agilemanifesto.org/iso/fr/manifesto.html

[2] AUBRY C., 2013. *Scrum: Le guide pratique de la méthode agile la plus populaire*. Dunod

[3] ROCHKIND M., 2014. *Programming Chrome Apps*. O'Reilly Media

[4] MEHTA P., 2016. *Creating Google Chrome Extensions*. Apress

9 Liste des figures

10 Liste des tableaux

Étude et réalisation d’un émulateur pour framework d’applications mobiles hybrides. Mémoire d'Ingénieur C.N.A.M., Rennes 2016.

Résumé

UXFME est un framework permettant de développer, packager et maintenir des applications mobiles hybrides, rendant accessibles des APIs natives à une application web. Pour compléter ce framework et faciliter le développement d’application, il a été demandé de fournir un émulateur d’appareils mobiles (smartphones, tablettes) pour ce framework d’applications mobile hybrides.

Après étude des solutions préexistantes, il s’est révélé qu’aucune ne satisfaisait complètement aux critères et qu’il était donc nécessaire de développer cette solution. Après une phase d’évaluation technique il était possible de concevoir une architecture logicielle.
Enfin, il a fallu répondre durant la réalisation à certaines difficultés techniques intrinsèques à l’émulation mobile.

Mots clés : Mobile, émulation, application hybride, extension Chrome

---

Summary

UXFME is a framework for developing, packaging and maintaining hybrid mobile applications, giving access to native APIs to a web application. To complete this framework and ease application development, it has been asked to provide an emulator of mobile devices (smartphones and tablets) for this hybrid mobile applications framework.

After a study of the existing solutions, it appeared that none fully met the criteria and that it was necessary to develop this solution. It was possible to design a software architecture after a technical evaluation phase.
Finally, it was necessary during project execution to solve some technical difficulties due to the nature of mobile emulation.

Keywords : Mobile, emulation, hybrid application, Chrome extension